

Internet of Things for Architects

Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security



Packt

www.packt.com

By Perry Lea

Internet of Things for Architects

Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security

Perry Lea

Packt

BIRMINGHAM - MUMBAI

Internet of Things for Architects

Copyright © 2018 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Gebin George

Acquisition Editor: Shrilekha Inani

Content Development Editor: Sharon Raj

Technical Editors: Prashant Chaudhari, Komal Karne

Copy Editor: Safis Editing

Project Coordinator: Virginia Dias

Proofreader: Safis Editing

Indexer: Aishwarya Gangawane

Graphics: Tania Dutta

Production Coordinator: Shantanu Zagade

First published: January 2018

Production reference: 1190118

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

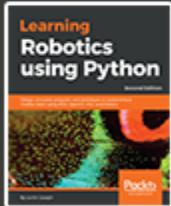
B3 2PB, UK.

ISBN 978-1-78847-059-9

www.packtpub.com



GET
80%
OFF ANY PACKT
EBOOK AND VIDEO



Go to www.packtpub.com
and use this code in the checkout:

HBIOT80OFF

Packt >



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

PacktPub.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Perry Lea has spent 21 years at Hewlett Packard as a distinguished technologist and chief architect. He then served as a distinguished member of technical staff and strategic director at Micron Technologies leading a team working on advanced computing devices. He is currently a technical director at Cradlepoint where he leads advancement and research in IoT and fog compute.

Perry has degrees in computer science, computer engineering, and an EE degree from Columbia University. He is a senior member of IEEE and a senior member/distinguished speaker of ACM. He has 8 patents with 40 pending.

Thanks to my wife, Dawn, and family and friends for being the support team to complete this book.

I wish to thank Sandra Capri of Ambient Sensors for critical review and comment on sensors and near-range communication.

I also thank David Rush from Cradlepoint for the comment on long-range connectivity and cellular systems.

Finally, a thank you to the numerous consortiums and technical communities, such as IEEE and ACM.

About the reviewer

Parkash Karki is a principal architect and product development manager with over 20 years of experience in the IT field. With a BSc (Hons) physics from the University of Delhi and master of computer applications from BIAS, he is PMP certified and also holds other certifications in Microsoft technologies. His has majorly worked on various Microsoft and open source technologies with vast experience in DevOps and Azure Cloud. As a DevOps and Cloud architect, he helps his customers adopt them well. He is very passionate about IoT, artificial intelligence, and automation technologies.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface	1
Chapter 1: The IoT Story	7
History of the IoT	11
IoT potential	13
Industrial and manufacturing	16
Industrial and manufacturing IoT use cases and impact	17
Consumer	17
Consumer IoT use cases	18
Retail, financial, and marketing	18
Retail IoT use cases	18
Healthcare	19
Healthcare IoT use cases	19
Transportation and logistics	20
Transportation and logistics IoT use cases	20
Agricultural and environmental	21
Agricultural and environmental IoT use cases	21
Energy	22
Energy IoT use cases	22
Smart city	22
Smart city IoT use cases	23
Government and military	24
Government and military IoT use cases	24
Summary	25
Chapter 2: IoT Architecture and Core IoT Modules	26
IoT ecosystem	27
IoT versus machine to machine	28
The value of a network and Metcalfe's and Beckstrom's law	29
IoT architecture	32
Role of an architect	33
Part 1 – Sensing and power	34
Part 2 – Data communication	35
Part 3 – Internet routing and protocols	36
Part 4 – Fog and edge compute, analytics, and machine learning	36
Part 5 – Threat and security in IoT	37
Summary	38

Chapter 3: Sensors, Endpoints, and Power Systems	39
Sensing devices	40
Thermocouples and temperature sensing	40
Thermocouples	40
Resistance Temperature Detectors	42
Thermistors	43
Temperature sensor summary	44
Hall effect sensors and current sensors	44
Photoelectric sensors	46
PIR sensors	46
LiDAR and active sensing systems	48
MEMS sensors	50
MEMS accelerometers and gyroscopes	50
MEMS microphones	53
MEMS pressure sensors	54
Smart IoT endpoints	55
Vision system	56
Sensor fusion	59
Input devices	60
Output devices	60
Functional examples (putting it all together)	61
Functional example – TI SensorTag CC2650	61
Sensor to controller	64
Energy sources and power management	65
Power management	66
Energy harvesting	67
Solar harvesting	69
Piezo-mechanical harvesting	70
RF energy harvesting	71
Thermal harvesting	72
Energy storage	73
Energy and power models	73
Batteries	77
Supercapacitors	77
Radioactive power sources	78
Energy storage summary and other forms of power	78
Summary	79
Chapter 4: Communications and Information Theory	80
Communication theory	82
RF energy and theoretical range	82
RF interference	87

Information theory	88
Bitrate limits and the Shannon-Hartley theorem	89
Bit error rate	94
Narrowband versus wideband communication	97
The radio spectrum	101
Governing structure	101
Summary	105
Chapter 5: Non-IP Based WPAN	106
Wireless personal area network standards	107
802.15 standards	107
Bluetooth	109
Bluetooth history	109
Bluetooth 5 communication process and topologies	111
Bluetooth 5 stack	113
Bluetooth 5 PHY and interference	117
Bluetooth packet structure	119
BR/EDR operation	121
BLE operation	123
Bluetooth profiles	127
BR/EDR security	129
BLE security	130
Beaconing	131
Bluetooth 5 range and speed enhancement	137
Bluetooth mesh introduction	139
Bluetooth mesh topology	141
Bluetooth mesh addressing modes	144
Bluetooth mesh provisioning	146
IEEE 802.15.4	147
IEEE 802.15.4 architecture	147
IEEE 802.15.4 topology	152
IEEE 802.15.4 address modes and packet structure	153
IEEE 802.15.4 start-up sequence	154
IEEE 802.15.4 security	155
Zigbee	157
Zigbee history	157
Zigbee overview	158
Zigbee PHY and MAC (and difference from IEEE 802.15.4)	160
Zigbee protocol stack	161
Zigbee addressing and packet structure	162
Zigbee mesh routing	164
Zigbee association	165
Zigbee security	165
Z-Wave	167
Z-Wave overview	167

Z-Wave protocol stack	170
Z-Wave addressing	171
Z-Wave topology and routing	172
Summary	174
Chapter 6: IP-Based WPAN and WLAN	175
Internet protocol and transmission control protocol	176
IP role in IoT	176
WPAN with IP – 6LoWPAN	179
6LoWPAN topology	180
6LoWPAN protocol stack	182
Mesh addressing and routing	183
Header compression and fragmentation	185
Neighbor discovery	188
6LoWPAN security	189
WPAN with IP – Thread	190
Thread architecture and topology	191
Thread protocol stack	192
Thread routing	193
Thread addressing	194
Neighbor discovery	195
IEEE 802.11 protocols and WLAN	196
IEEE 802.11 suite of protocols and comparison	197
IEEE 802.11 architecture	198
IEEE 802.11 spectrum allocation	200
IEEE 802.11 modulation and encoding techniques	202
IEEE 802.11 MIMO	207
IEEE 802.11 packet structure	210
IEEE 802.11 operation	213
IEEE 802.11 security	215
IEEE 802.11ac	216
IEEE 802.11p vehicle-to-vehicle	218
IEEE 802.11ah	222
Summary	227
Chapter 7: Long-Range Communication Systems and Protocols (WAN)	229
Cellular connectivity	230
Governance models and standards	231
Cellular access technologies	235
3GPP user equipment categories	236

4G-LTE spectrum allocation and bands	238
4G-LTE topology and architecture	242
4G-LTE E-UTRAN protocol stack	246
4G-LTE geographical areas, dataflow, and handover procedures	248
4G-LTE packet structure	251
Cat 0, Cat 1, Cat M1, and NB-IoT	252
LTE Cat-0	252
LTE Cat-1	253
LTE Cat-M1 (eMTC)	254
LTE Cat-NB	256
5G	258
LoRa and LoRaWAN	263
LoRa physical layer	264
LoRaWAN MAC layer	266
LoRaWAN topology	268
LoRaWAN summary	269
Sigfox	270
Sigfox physical layer	271
Sigfox MAC layer	273
Sigfox protocol stack	274
Sigfox topology	275
Summary	277
Chapter 8: Routers and Gateways	279
Routing functions	280
Gateway functions	280
Routing	280
Failover and out-of-band management	285
VLAN	286
VPN	288
Traffic shaping and QoS	290
Security functions	293
Metrics and analytics	294
Edge processing	295
Software-Defined Networking	296
SDN architecture	297
Traditional internetworking	299
SDN benefits	300
Summary	301
Chapter 9: IoT Edge to Cloud Protocols	302

Protocols	302
MQTT	305
MQTT publish-subscribe	306
MQTT architecture details	308
MQTT packet structure	310
MQTT communication formats	311
MQTT working example	315
MQTT-SN	317
MQTT-SN architecture and topology	318
Transparent and aggregating gateways	319
Gateway advertisement and discovery	320
Differences between MQTT and MQTT-SN	321
Constrained Application Protocol	321
CoAP architecture details	322
CoAP Messaging Formats	325
CoAP usage example	330
Other protocols	332
STOMP	332
AMQP	332
Protocol summary and comparison	335
Summary	337
Chapter 10: Cloud and Fog Topologies	338
Cloud services model	339
NaaS	340
SaaS	341
PaaS	341
IaaS	341
Public, private, and hybrid cloud	342
Private cloud	342
Public cloud	343
Hybrid cloud	343
The OpenStack cloud architecture	343
Keystone – identity and service management	345
Glance – image service	345
Nova compute	346
Swift – Object Storage	348
Neutron – Networking services	348
Cinder – Block Storage	349

Horizon	349
Heat – orchestration (optional)	350
Ceilometer – telemetry (optional)	350
Constraints of cloud architectures for IoT	351
Latency effect	352
Fog computing	354
The Hadoop philosophy for Fog computing	354
Fog Computing versus Edge Computing versus cloud computing	355
OpenFog Reference Architecture	356
Application services	357
Application support	358
Node management and software backplane	358
Hardware virtualization	359
OpenFog node security	359
Network	360
Accelerators	360
Compute	361
Storage	361
Hardware platform infrastructure	361
Protocol abstraction	362
Sensors, actuators, and control systems	362
Amazon Greengrass and Lambda	362
Fog Topologies	365
Summary	371
Chapter 11: Data Analytics and Machine Learning in the Cloud and in the Fog	372
Basic data analytics in IoT	373
Top-level cloud pipeline	376
Rules engines	378
Ingestion – streaming, processing, and data lakes	381
Complex event processing	384
Lambda architecture	385
Sector use cases	386
Machine learning in IoT	387
Machine learning models	392
Classification	393
Regression	396
Random forest	396
Bayesian models	398
Convolutional Neural Networks	401
First layer and filters	402

Max pooling and subsampling	402
Hidden layers and formal description on forwarding propagation	403
CNN examples	405
CNN training and backpropagation	407
RNN	411
Training and inference for IoT	418
IoT data analytics and machine learning comparison and assessment	419
Summary	421
Chapter 12: IoT Security	422
Cyber security vernacular	423
Attack and threat terms	423
Defense terms	425
Anatomy of IoT cyber attacks	427
Mirai	428
Stuxnet	430
Chain Reaction	431
Physical and hardware security	433
Root of Trust	433
Key management and trusted platform modules	435
Processor and memory space	436
Storage security	436
Physical security	437
Cryptography	439
Symmetric cryptography	441
Asymmetric cryptography	445
Cryptographic hash (authentication and signing)	449
Public Key Infrastructure	450
Network stack – Transport Layer Security	452
Software defined perimeter	454
Software-Defined Perimeter architecture	454
Blockchains and cryptocurrencies in IoT	456
Bitcoin (blockchain-based)	457
IOTA (directed acyclical graph-based)	462
Government regulations and intervention	463
US Congressional Bill –Internet of Things (IoT) Cybersecurity Improvement Act of 2017	463
Other governmental bodies	465
IoT security best practices	466
Holistic security	466

Security checklist	467
Summary	468
Chapter 13: Consortiums and Communities	469
PAN consortia	469
Bluetooth SIG	470
Thread Group	470
Zigbee Alliance	471
Miscellaneous	471
Protocol consortia	471
Open Connectivity Foundation and Allseen Alliance	472
OASIS	472
Object Management Group	473
IPSO Alliance	474
Miscellaneous	474
WAN consortia	474
Weightless	475
LoRa Alliance	475
Internet Engineering Task Force	475
Wi-Fi Alliance	476
Fog and edge consortia	477
OpenFog	477
EdgeX Foundry	477
Umbrella organizations	478
Industrial Internet Consortium	478
IEEE IoT	479
Miscellaneous	479
US government IoT and security entities	480
Summary	480
Appendix A: Other Books You May Enjoy	481
Leave a review - let other readers know what you think	483
Index	484

Preface

You probably experience the Internet of Things on a daily basis in your personal and work life. Much of the public's impression of IoT is from their personal interaction with a Fitbit fitness tracker, an Amazon Echo assistant, or a Google thermostat.

A 2017 search for the keyword IoT on LinkedIn reveals 7,189 job postings related to IoT. Glassdoor shows 5,440 and <http://monster.com/> shows more than a thousand requisitions. The IoT market is booming for talent as well as solutions. As is often the case, technologists will take a path of least resistance to binding what had been an unconnected object to the internet. That approach certainly works, but it is different than the role of an architect. An architect needs to understand the big picture of disparate technologies, scaling factors, security, and energy to build an IoT solution that not only works but provides value to their company, customers, and shareholders.

Many IoT projects fail or are stuck in R&D for two reasons. First, building a robust system is difficult from a security and robustness perspective. Second, often is the case that an IoT solution technically works, but it is not manageable from the perspective of the purchasing IT manager. As we place more things on the internet, we as architects need to consider the enterprise and industrial IT world is a 50-year-old mature industry. Placing an IP address on a lightbulb is certainly possible, but not necessarily manageable from the customer perspective. This book attempts to address the IoT from an enterprise/industrial/commercial perspective rather than a hobbyist perspective.

This book covers IoT from an architectural and holistic point of view from sensor to cloud including all the physical transports and transformation between the two. Because this book is an architectural guide, it attempts to maintain enough depth to teach another architect the constraints and discipline of an underlying system. There are countless books and tutorials on IoT specifics, such as MQTT protocol, cloud design and DevOps, power and battery design, and RF signal analysis. These are all important components for an IoT system, and a qualified architect should be able to span the breadth to design a robust system. However, the architect must understand when to pull up from design details to continue to provide value as an architect.

It isn't expected that a reader come to this book with an inherent knowledge of every engineering domain. This book touches on radio frequency signaling, power and energy, and circuit theory. On the other side of the aisle, the book goes into internet protocol programming and cloud provisioning. Finally, it will dive deep into machine learning applications such as convolutional neural networks. Having all the skills to bring these technologies together is an architect's function. This book helps you get to that level, but it doesn't expect you to come with a deep understanding of each science.

What you can do with IoT is incredible as it will usher in the next major revolution in manufacturing, healthcare, government, and enterprise. It will have major impact, yet inevitable, to the world GDP, employment, and markets. It also poses the greatest challenges and risk in security as you will learn.

Of those thousands of jobs listed, many are for IoT architects, technologists, and principals to build IoT solutions rather than widgets. This book will help you learn and apply technologies for those types of projects.

Additionally, it's fun. Designing a device for monitoring your home lighting or controlling thousands of streetlights in a city from the other side of the globe or on an airplane is a significantly powerful technology, made for techno-junkies but applied by architects.

Who this book is for

This book is aimed at for architects, system designers, technologists, and technology managers who want to understand the IoT ecosystem, various technologies, and trade-offs and develop a 50,000-foot view of IoT architecture.

What this book covers

Chapter 1, *The IoT Story*, introduces you to the growth, the importance, and the impact of the IoT from a narrative and historical perspective. You will also learn of use cases in various areas including industrial IoT, smart cities, transportation, and healthcare.

Chapter 2, *IoT Architecture and Core IoT Modules*, presents the overall picture of the combination of technologies covered in this book. Each segment has a purpose and can unknowingly affect each other. This is an important chapter for an architect to understand the "big picture" of inter-related technologies. This chapter also explores the ways to place a value on IoT.

Chapter 3, *Sensors, Endpoints, and Power Systems*, explores billions of edge endpoints and sensor technologies that will be placed on the internet. Fundamentals of sensor designs, architectures, and power systems are taught.

Chapter 4, *Communications and Information Theory*, will review important material on the dynamics and mathematical models that define communication systems important to IoT. You will understand the theory behind architectural decisions in selecting the proper forms of telecommunication.

Chapter 5, *Non-IP-Based WPAN*, discusses all the major non-IP-based protocols and technologies at the IoT Edge. This chapter includes a deep review of the new Bluetooth 5 architecture, Zigbee, Z-Wave, and mesh topologies for sensor networks.

Chapter 6, *IP-Based WPAN and WLAN*, will complete the near-range communication with a treatment of IP-based communication, including 6LoWPAN, Thread, and IEEE 802.11 standards. This chapter also details new 802.11 protocols such as 802.11p for vehicular communication and 802.11ah for IoT.

Chapter 7, *Long-Range Communication Systems and Protocols (WAN)*, covers wide area network and long-range communication transport data from things to the cloud. This chapter covers in detail all the cellular LTE standards, LoRaWAN, Sigfox, as well as new LTE narrowband and 5G architectures.

Chapter 8, *Routers and Gateways*, discusses the importance of edge routing and gateway functions. This chapter explores routing systems, gateway functions, VPNs, VLANs, and traffic shaping, and it covers software-defined networking.

Chapter 9, *IoT Edge to Cloud Protocols*, introduces you to the prevalent IoT to cloud protocols, such as MQTT, MQTT-SN, CoAP, AMQP, and STOMP. You will learn how to use them and, importantly, which to use.

Chapter 10, *Cloud and Fog Topologies*, explores the fundamentals of cloud architectures using OpenStack as a reference. You will learn of cloud constraints and how fog computing (using frameworks such as the OpenFog standard) seeks to solve these problems.

Chapter 11, *Data Analytics and Machine Learning in the Cloud and in the Fog*, covers the technologies and use cases for analyzing the myriad of IoT data efficiently using tools, such as rules engines, complex event processing, and lambdas. This chapter also explores machine learning applications for IoT data and where it makes sense to use them.

Chapter 12, *IoT Security*, covers security from a holistic view for every IoT component covered in this book. You will understand the theory and architecture of protocol, hardware, software-defined perimeter, and block-chain security.

Chapter 13, *Consortiums and Communities*, details the numerous industrial, academic, and government consortiums defining the standards and rules around the Internet of Things.

To get the most out of this book

There are several examples of hardware design and coding examples in this book. Most of the coding examples are pseudo-code based on Python syntax. Working examples are also based on Python 3.4.3 that is usable on Mac OS X, Linux, and Microsoft. In areas (such as Chapter 9, *IoT Edge to Cloud Protocols*, libraries such as MQTT (such as Paho) are freely available for use in Python.

Having familiarity with some foundational calculus, information theory, electrical properties, and computer science can only help us gain a deeper insight into IoT from an architectural perspective.

Some examples show scripting within Chapter 10, *Cloud and Fog Topologies*, use OpenStack or Amazon AWS/Greengrass. In those cases, acquiring a cloud account is helpful but not strictly needed to understand the architectural goals.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it from https://www.packtpub.com/sites/default/files/downloads/InternetofThingsforArchitects_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The `insert` operation places a modification in the working memory."

A block of code is set as follows:

```
rule "Furnace_On"
when
Smoke_Sensor(value > 0) && Heat_Sensor(value > 0)
then
insert(Furnace_On())
end
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
rule "Furnace_On"
when
Smoke_Sensor(value > 0) && Heat_Sensor(value > 0)
then
insert(Furnace_On())
end
```

Any command-line input or output is written as follows:

```
aws greengrass create-function-definition --name "sensorDefinition"
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "**Internet Key Exchange (IKE)** is the security protocol in IPsec."

Warnings or important notes appear like this.



Tips and tricks appear like this.



Get in touch

Feedback from our readers is always welcome.

General feedback: Email feedback@packtpub.com and mention the book title in the subject of your message. If you have questions about any aspect of this book, please email us at questions@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packtpub.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packtpub.com.

1

The IoT Story

You wake up Tuesday, May 17, 2022, around 6:30 AM PST, as you always do. You never really needed an alarm clock, you are one of those types with some form of physiological clock. Immediately after, your eyes open to a fantastic sunny morning as it's approaching 70° C outside. You will take part in a day that will be completely different than the morning of Wednesday, May 17, 2017. Everything about your day, your lifestyle, your health, your finances, your work, your commute, even your parking spot will be different. Everything about the world you live in will be different: energy, healthcare, farming, manufacturing, logistics, mass transit, environment, security, shopping, and even clothing. This is the impact of connecting ordinary objects to the Internet, or the **Internet of Things (IoT)**. I think a better analogy is the Internet of Everything.

Before you even awakened, a lot has happened in the IoT that surrounds you. Your sleep behavior has been monitored by a sleep sensor or smart pillow. Data was sent to an IoT gateway and then streamed to a cloud service you use for free that reports to a dashboard on your phone. You don't need an alarm clock, but if you had another 5 A.M. flight you would set it—again, controlled by a cloud agent using **if this, then that (IFTTT)** protocol. Your dual zone furnace is connected to a different cloud provider and is on your home 802.11 Wi-Fi, as are your smoke alarms, doorbell, irrigation systems, garage door, surveillance cameras, and security system. Your dog is chipped with a proximity sensor using an energy harvesting source that lets him open the doggy door and tell you where he is.

You don't really have a PC anymore. You certainly have a tablet-style computer and a smartphone as your central *creation* device, but your world is based on using VR/AR Goggles since the screen is so much better and larger. You do have a fog computing gateway in your closet. It's connected to a 5G service provider to get you on the Internet and WAN because wired connections don't work for your lifestyle—you are mobile, connected, and online no matter where you are, and 5G and your favorite carrier make sure your experience is great in a hotel room in Miami or your home in Boise, Idaho. The gateway also performs a lot of actions in your home for you, such as processing video streams from those webcams to detect if there's been a fall or an accident in the house. The security system is being scanned for anomalies (strange noises, possible water leaks, lights being left on, your dog chewing on the furniture again). The edge node also acts as your home hub, backing up your phone daily because you have a tendency to break them, and serves as your private cloud even though you know nothing about cloud services.

You ride your bike to the office. Your bike jersey uses printable sensors, and monitors your heart rate and temperature. That data is streamed over Bluetooth Low Energy to your smartphone simultaneously while you listen to Bluetooth audio streamed from your phone to your Bluetooth earphones. On the way there, you pass several billboards all displaying video and real-time ads. You stop at your local coffee shop and there is a digital signage display out front calling you out by name and asking if you want the last thing you ordered yesterday: *a 12 oz Americano with room for cream*. It did this by a beacon and gateway recognizing your presence within 5 feet and approaching the display. You select *yes*, of course. Most people arrive at work via their car and are directed to the optimal parking space via smart sensors in each parking slot. You, of course, get the optimal parking space right out front with the rest of the cyclists.

Your office is part of a green energy program. Corporate mandated policies on a zero-emission office space. Each room has proximity sensors to detect not only if a room is occupied, but who is in the room. Your name badge to get in the office is a beaconing device on a 10-year battery. Your presence is known once you enter the door. Lights, HVAC, automated shades, ceiling fans, even digital signage is connected. A central fog node monitors all the building information and syncs it to a cloud host. A rules engine has been implemented to make real-time decisions based on occupancy, time of day, and the season of the year, as well as inside and outside temperatures. Environmental conditions are ramped up or down to maximize energy utilization. There are even sensors on the main breakers *listening* to the patterns of energy and making a decision on the fog nodes if there are strange patterns of energy usage that need examination.

It does all this with several real-time streaming edge analytics and machine learning algorithms that have been trained on the cloud and pushed to the edge. The office hosts a 5G small cell to communicate externally to the upstream carrier, but they also host a number of small-cell gateways internally to focus signals within the confines of the building. The internal 5G acts as a LAN as well.

Your phone and tablet have switched to the internal 5G signal, and you switch on your software-defined network overlay and are instantly on the corporate LAN. Your smartphone does a lot of work for you; it is essentially your personal gateway to your own personal area network surrounding your body. You drop into your first meeting today, but your co-worker isn't there and arrives a few minutes late. He apologizes, but explains his drive to work was eventful. His newer car informed the manufacturer of a pattern of anomalies in the compressor and turbocharger. The manufacturer was immediately informed of this and called the owner to inform him that the vehicle has a 70 percent chance of having a failed turbo within two days of his typical commute. They scheduled an appointment with the dealership, and have the new parts arriving to fix the compressor. This saved him considerable cost in replacing the turbo and a lot of aggravation.

For lunch, the team decides to go out to a new fish taco place downtown. A group of four of you manage your way into a coupe more comfortable for two than four and make your way. Unfortunately, you'll have to park in one of the more expensive parking structures. Parking rates are dynamic and follow a supply and demand basis. Because of some events and how full the lots are, the rates doubled even for mid-day Tuesday. On the bright side, the same systems raising the parking fees also inform your car and smartphone exactly which lots and which space to drive to. You punch in the fish taco address, the lot and capacity pop up, and you reserve a spot before you arrive. The car approaches the gate, which identifies your phone signature and opens up. You drive to the spot and the application registers with the parking cloud that you are in the right spot over the correct sensor.

That afternoon, you need to go to the manufacturing site on the other side of town. It's a typical factory environment: several injection molding machines, pick-and-place devices, packaging machines, and all the supporting infrastructure. Recently, the quality of the product has been slipping. The final product has joint connection problems and is cosmetically inferior to last month's lot. After arriving at the site, you talk to the manager and inspect the site. Everything appears normal, but the quality certainly has been marginalized. The two of you meet and bring up the dashboards of the factory floor.

The system uses a number of sensors (vibration, temperature, speed, vision, and tracking beacons) to monitor the floor. The data is accumulated and visualized in real time. There are a number of predictive maintenance algorithms watching the various devices for signs of wear and error. That information is streamed to the equipment manufacturer and your team as well. The logs and trend analysis didn't pick up any abnormal behavior, and had been trained by your best experts. This looks like the type of problem that would turn hours into weeks and force the best and brightest in your organization to attend expensive daily SWOT team meetings. However, you have a lot of data. All the data from the factory floor is preserved in a long-term storage database. There was a cost to that service, and at first it was difficult to justify, but you think it may have paid for itself a thousandfold. Taking all that historical data through a complex event processor and analytics package, you quickly develop a set of rules that model the quality of your failing parts. Working backward to the events that led to the failures, you realize it is not a point failure, but several aspects:

- The internal temperature of the working space rose 2° C to conserve energy for the summer months
- The assembly slowed down output by 1.5 percent due to supply issues
- One of the molding machines was nearing a predictive maintenance period and the temperature and assembly speed pushed its failing case over the predicted value

You found the issue, and retrained the predictive maintenance models with the new parameters to catch this case in the future. Overall, not a bad day at work.

While this fictional case may or may not be true, it's pretty close to reality today. The IoT is defined by Wikipedia: https://en.wikipedia.org/wiki/Internet_of_things as *The Internet of things (IoT) is the inter-networking of physical devices, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity which enable these objects to collect and exchange data.*

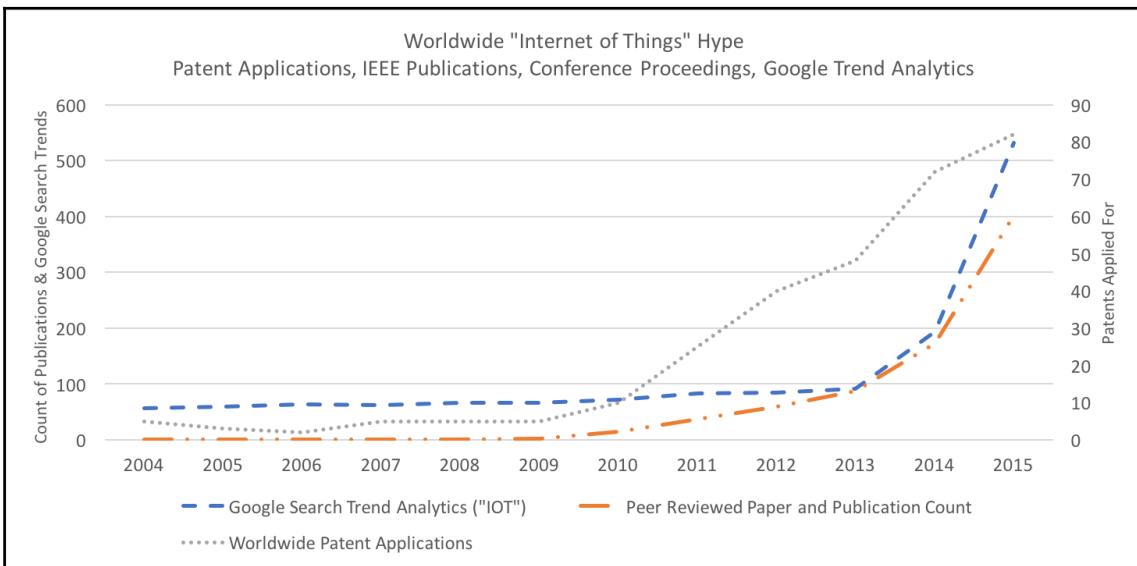
History of the IoT

The term IoT can most likely be attributed to Kevin Ashton in 1997 with his work at Proctor and Gamble using RFID tags to manage supply chains. The work brought him to MIT in 1999 where he and a group of like-minded individuals started the Auto-ID center research consortium (for more information, visit <http://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/>). Since then, IoT has taken off from simple RFID tags to an ecosystem and industry that by 2020 will cannibalize, create, or displace five trillion out of one hundred trillion global GDP dollars, or 6% of the world GDP. The concept of things being connected to the Internet up through 2012 was primarily connected smartphones, tablets, PCs, and laptops. Essentially, things that first functioned in all respects as a computer. Since the humble beginnings of the Internet starting with ARPANET in 1969, most of the technologies surrounding the IoT didn't exist. Up to the year 2000, most devices that were associated with the Internet were, as stated, computers of various sizes. The following timeline shows the slow progress in connecting *things* to the Internet:

Year	Device	Reference
1973	Mario W. Cardullo receives the patent for first RFID tag	US Patent US 3713148 A
1982	Carnegie Mellon internet-connected soda machine	https://www.cs.cmu.edu/~coke/history_long.txt
1989	Internet-connected toaster at Interop '89	IEEE Consumer Electronics Magazine (Volume: 6, Issue: 1, Jan. 2017)
1991	HP introduces HP LaserJet IISi: first Ethernet-connected network printer	http://hpmuseum.net/display_item.php?hw=350
1993	Internet-connected coffee pot at University of Cambridge (first internet-connected camera)	https://www.cl.cam.ac.uk/coffee/qsf/coffee.html
1996	General Motors OnStar (2001 remote diagnostics)	https://en.wikipedia.org/wiki/OnStar
1998	Bluetooth SIG formed	https://www.bluetooth.com/about-us/our-history
1999	LG Internet Digital DIOS refrigerator	https://www.telecompaper.com/news/lg-unveils-internetready-refrigerator--221266

2000	First instances of <i>Cooltown</i> concept of pervasive computing everywhere: HP Labs, a system of computing and communication technologies that, combined, create a <i>web-connected experience for people, places, and objects</i>	https://www.youtube.com/watch?v=U2AkkuIVV-I
2001	First Bluetooth product launched: KDDI Bluetooth-enabled mobile phone	http://edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html
2005	United Nation's International Telecommunications Union report predicting the rise of IoT for the first time	http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf
2008	IPSO Alliance formed to promote IP on objects, first IoT-focused alliance	https://www.ipso-alliance.org
2010	The concept of Smart Lighting formed after success in developing solid-state LED light bulbs	https://www.bu.edu/smartlighting/files/2010/01/BobK.pdf
2014	Apple creates iBeacon protocol for beacons	https://support.apple.com/en-us/HT202880

Certainly, the term IoT has generated a lot of interest and hype. One can easily see that from a buzzword standpoint, the number of patents issued (<https://www.uspto.gov>) has grown exponentially since 2010. The number of Google searches (<https://trends.google.com/trends/>) and IEEE peer-reviewed paper publications hit the knee of the curve in 2013:



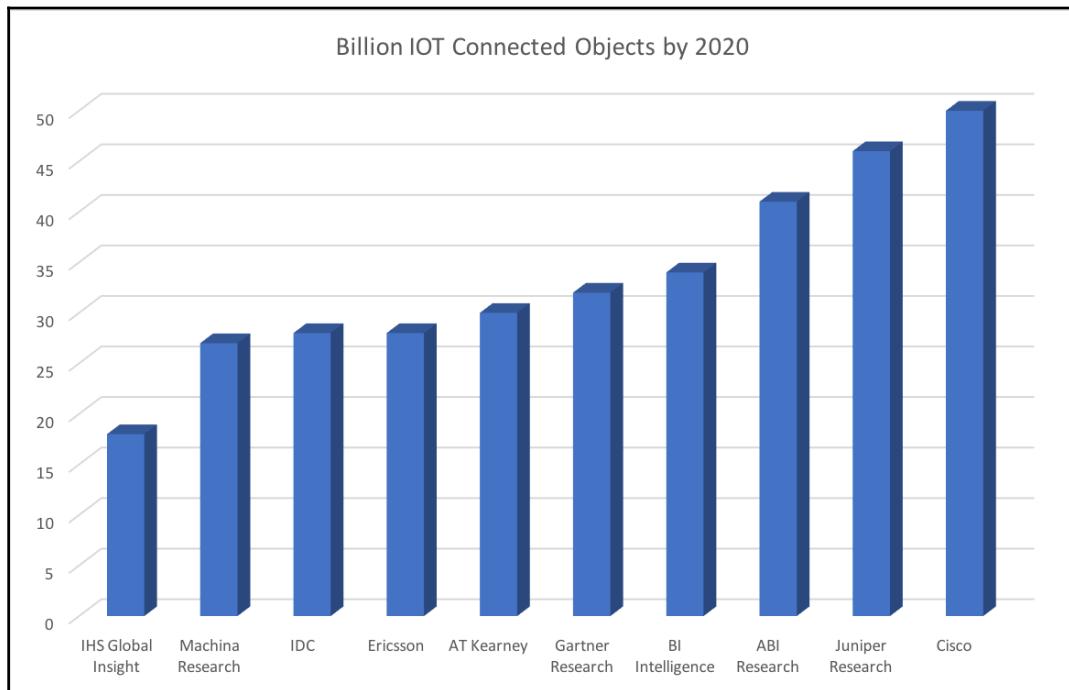
Analysis of keyword searches for IoT, patents, and technical publications

IoT potential

The IoT will touch nearly every segment in industrial, enterprise, health, and consumer products. It is important to understand the impact, as well as why these disparate industries will be forced to change in the way they build products and provide services. Perhaps your role as an architect forces you to focus on one particular segment; however, it is helpful to understand the overlap with other use cases.

As previously mentioned, there is an opinion that the impact of IoT-related industries, services, and trade will affect three percent (The route to a trillion devices, ARM Ltd 2017: https://community.arm.com/cfs-file/__key/telligent-evolution-components-attachments/01-1996-00-00-01-30-09/ARM_2D00_The-route-to-a-trillion-devices_2D00_June-2017.pdf) to four percent (The Internet of Things: Mapping Value Beyond the Hype, McKinsey and Company 2015: https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/The%20Internet%20of%20Things%20The%20value%20of%20digitizing%20the%20physical%20world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ashx) of global GDP by 2020 (extrapolated). Global GDP for 2016 was \$75.64 trillion dollars, with an estimate that by 2020 it will rise to \$81.5 trillion. That provides a range of value from IoT solutions of \$2.4 trillion to about \$4.9 trillion.

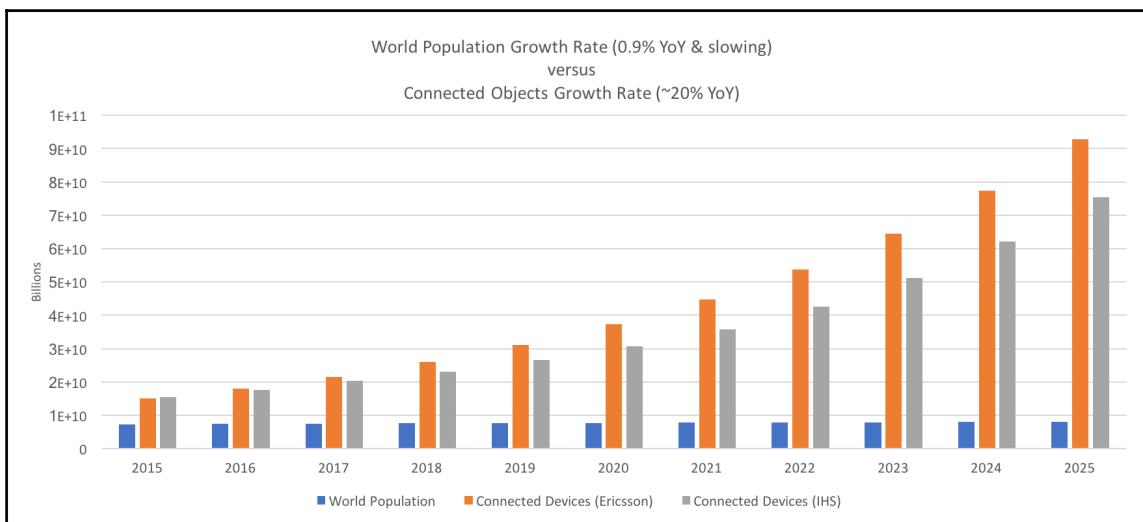
The scale of connected objects is unprecedented. Speculation of industry growth is imperiled with risks. To help normalize the impact, we look at several research firms and reports on the number of connected objects by the year 2020. The range is large, but still in the same order of magnitude. The average of these 10 analyst forecasts is about 33.4 billion connected things by 2020-2021. ARM recently conducted a study and forecast that by 2035 one trillion connected devices will be operational. By all accounts, the projects growth rate in the near term is about 20 percent year over year.



Different analyst and industry claims on the number of connected things

These numbers should impress the reader. For example, if we took a very conservative stance and predict that only 20 billion newly connected devices will be deployed (excluding the traditional computing and mobile products), we would be saying that 211 new Internet connected objects will come online every second.

Why this is of significance to the technology industry and IT sector is the fact that world population currently has a growth rate of roughly 0.9 percent to 1.09 percent per year (<https://esa.un.org/unpd/wpp/>). World population growth rate peaked in 1962 at 2.6 percent year over year, and has steadily been declining due to a number of factors. First and foremost, improvement in world GDP and economies has a propensity to reduce birth rates. Other factors include wars and famine. That growth implies that human-connected objects will plateau and machine to machine and connected objects will represent the majority of devices connected to the internet. This is important because the IT industry applies value to a network not necessarily by how much data is consumed, but by how many connections there are. This, generally speaking, is Metcalfe's law, and we will talk about that later in this book. It is also worth noting that after the first public website went live at CERN in 1990, it took 15 years to reach 1 billion people on Earth over the Internet. IoT is looking to add 6 billion connected devices per year. This, of course, is swaying the industry:



The disparity between human population growth versus connected thing growth. The trend has been a 20 percent growth of connected objects versus a nearly flat 0.9 percent human growth. Humans will no longer drive network and IT capacity.

It should be noted that economic impact is not solely revenue generation. The impact from IoT or any technology comes in the form of:

- New revenue streams (green energy solutions)
- Reducing costs (in-home patient healthcare)
- Reducing time to market (factory automation)
- Improving supply chain logistics (asset tracking)
- Reducing production loss (theft, spoilage of perishable)
- Increasing productivity (machine learning and data analytics)
- Cannibalization (Nest replacing traditional thermostats)

In our discussion throughout this book, it should be at the top of our minds as to what value an IoT solution delivers. If it is simply a new gadget, there will be a limited market scope. Only when the foreseeable benefit outweighs the cost will an industry thrive. Generally speaking, the target used should be a 5x improvement over a traditional technology. That has been my goal in the IT industry. When considering the cost of change, training, acquisition, support, and so on, a 5x differential is a fair rule of thumb.

We now detail the sectors of industry and how IoT will affect them.

Industrial and manufacturing

Industrial IoT (IIoT) is one of the fastest and largest segments in the overall IoT space by the number of connected things and the value those services bring to manufacturing and factory automation. This segment has traditionally been the world of operations technology (OT). This involves hardware and software tools to monitor physical devices. Traditional information technology roles have been administered differently than OT roles. OT will be concerned with yield metrics, uptime, real-time data collection and response, and systems safety. The IT role will concentrate on security, groupings, data delivery, and services. As the IoT becomes prevalent in industry and manufacturing, these worlds will combine especially with predictive maintenance from thousands of factory and production machines to deliver an unprecedented amount of data to private and public cloud infrastructure.

Some of the characteristics of this segment include the need to provide near real-time or at real-time decisions for OT. This means latency is a major issue for IoT on a factory floor. Additionally, downtime and security are a top concern. This implies the need for redundancy, and possibly private cloud networks and data storage. The industrial segment is one of the fastest-growing markets. One nuance of this industry is the reliance of *brownfield* technology, meaning hardware and software interfaces that are not mainstream. It is often the case that 30-year-old production machines rely on RS485 serial interfaces rather than modern wireless mesh fabrics.

Industrial and manufacturing IoT use cases and impact

Following are the industrial and manufacturing IoT use cases and their impact:

- Preventative maintenance on new and pre-existing factory machinery
- Throughput increase through real-time demand
- Energy savings
- Safety systems such as thermal sensing, pressure sensing, and gas leaks
- Factory floor expert systems

Consumer

Consumer-based devices were one of the first segments to adopt things being connected on the internet. Consumer IoT came into form as a connected coffee pot at a university in the 1990s. It flourished with the adoption of Bluetooth for consumer use in the early 2000s. Now millions of homes that have Nest thermostats, Hue lightbulbs, Alexa assistants, and Roku set-top boxes. People too are connected with Fitbits and other wearable technology. The consumer market is usually first to adopt these new technologies. We can also think of these as gadgets. All are neatly packaged and wrapped devices that are essentially plug and play.

One of the constraints in the consumer market is the bifurcation of standards. We see, for example, several WPAN protocols have a footing like Bluetooth, Zigbee, and Z-wave (all being non-interoperable).

This segment also has common traits in the healthcare market, with wearable devices and home health monitors. We keep them separate for this discussion, and healthcare will grow beyond simple connected home health devices (for example, beyond the functionality of a Fitbit).

Consumer IoT use cases

The following are some of the consumer IoT use cases:

- **Smart home gadgetry:** Smart irrigation, smart garage doors, smart locks, smart lights, smart thermostats, and smart security.
- **Wearables:** Health and movement trackers, smart clothing/wearables.
- **Pets:** Pet location systems, smart dog doors.

Retail, financial, and marketing

This category refers to any space where consumer-based commerce transacts. This can be a brick and mortar store or a pop-up kiosk. Additionally, this category refers to why we include financial institutions and marketing fields in this area. These include traditional banking services and insurers, but also leisure and hospitality services. Retail IoT impact is already in process, with the goal of lowering sales costs and improving customer experience. This is done with a myriad of IoT tools. For simplicity in this book, we also add advertising and marketing to this category.

This segment measures value in immediate financial transactions. If the IoT solution is not providing that response, its investment must be scrutinized. This drives constraints on finding new ways to either save costs, or drive revenue. Allowing customers to be more efficient allows retailers and service industries to move customers quickly, and to do so with less staffing resources.

Retail IoT use cases

Some of the retail IoT use cases are as follows:

- Targeted advertising, such as locating known or potential customers by proximity and providing sales information.

- Beaconing, such as proximity sensing customers, traffic patterns, and inter-arrival times as marketing analytics.
- Asset tracking, such as inventory control, loss control, and supply chain optimizations.
- Cold storage monitoring, such as analyze cold storage of perishable inventory. Apply predictive analytics to food supply.
- Insurance tracking of assets.
- Insurance risk measurement of drivers.
- Digital signage within retail, hospitality, or citywide.
- Beaconing systems within entertainment venues, conferences, concerts, amusement parks, and museums.

Healthcare

The healthcare industry will contend with industrial and logistics for the top spot in revenue and impact on IoT. Any and all systems that improve the quality of life and reduce health costs is a top concern in nearly every developed country. The IoT is poised to allow for remote and flexible monitoring of patients wherever they may be. Advanced analytics and machine learning tools will observe patients in order to diagnose illness and prescribe treatments. Such systems will also be the watchdogs in the event of needed life-critical care. Currently, there are about 500 million wearable health monitors, with double-digit growth in the years to come.

The constraints on healthcare systems are significant. From HIPAA compliance to the security of data, IoT systems need to act like hospital quality tools and equipment. Field systems need to communicate with healthcare centers 24/7, reliably and with zero downtime if the patient is being monitored at home. Systems may need to be on a hospital network while monitoring a patient in an emergency vehicle.

Healthcare IoT use cases

Some of the healthcare IoT use cases are as follows:

- In-home patient care
- Learning models of predictive and preventative healthcare
- Dementia and elderly care and tracking

- Hospital equipment and supply asset tracking
- Pharmaceutical tracking and security
- Remote field medicine
- Drug research
- Patient fall indicators

Transportation and logistics

Transportation and logistics will be significant, if not the leading driver in IoT. The use cases involve tracking the asset on devices being delivered, transported, or shipped, whether that's on a truck, train, plane, or boat. This is also the area of connected vehicles that communicate to offer assistance to the driver, or preventative maintenance on behalf of the driver. Right now, an average vehicle purchased new off a lot will have about 100 sensors. That number will double as vehicle-to-vehicle communication, vehicle-to-road communication, and automated driving become must-have features for safety or comfort. This has important roles beyond consumer vehicles, and extends to rail lines and shipping fleets that cannot absorb any downtime. We will also see service trucks that can track assets within a service vehicle. Some of the use cases can be very simple, but also very costly, such as monitoring the location of service vehicles in the delivery of stock. Systems are needed to automatically route trucks and service personnel to locations based on demand versus routine.

This mobile-type category has the requirement of geolocation awareness. Much of this comes from GPS navigation. From an IoT perspective, the data analyzed would include assets and time, but also spatial coordinates.

Transportation and logistics IoT use cases

Following are some of the transportation and logistics IoT use cases:

- Fleet tracking and location awareness
- Railcar identification and tracking
- Asset and package tracking within fleets
- Preventative maintenance of vehicles on the road

Agricultural and environmental

Farming and environmental IoT includes elements of livestock health, land and soil analysis, micro-climate predictions, efficient water usage, and even disaster predictions in the case of geological and weather-related disasters. Even as the world population growth slows down, world economies are becoming more affluent. Hunger and starvation crises are rare. That said, the demand for food production is set to double by 2035. Significant efficiencies in agriculture can be achieved through IoT. Using smart lighting to adjust the spectrum frequency based on poultry age can increase growth rates and decrease mortality rates based on stress on chicken farms. Additionally, smart lighting systems could save \$1 billion annually on energy versus the common dumb incandescent lighting currently used. Other uses include detecting livestock health based on sensor movement and positioning. A cattle farm could find animals with the propensity of sickness before a bacterial or viral infection were to spread. Edge analysis systems could find, locate, and isolate heads of cattle in real time, using data analytics or machine learning approaches.

This segment also has the distinction of being in remote areas (volcanoes) or sparse population centers (corn field). This has impacts on data communication systems that we will need to consider in later *Chapter 5, Non-IP Based WPAN* and *Chapter 7, Long-Range Communication Systems and Protocols (WAN)*.

Agricultural and environmental IoT use cases

Some of the agricultural and environmental IoT use cases are as follows:

- Smart irrigation and fertilization techniques to improve yield
- Smart lighting in nesting or poultry farming to improve yield
- Livestock health and asset tracking
- Preventative maintenance on remote farming equipment via manufacturer
- Drones-based land surveys
- Farm-to-market supply chain efficiencies with asset tracking
- Robotic farming
- Volcanic and fault line monitoring for predictive disasters

Energy

The energy segment includes the monitoring of energy production at source to and through the usage energy at the client. A significant amount of research and development has focused on consumer and commercial energy monitors such as smart electric meters that communicate over low-power and long-range protocols to reveal real-time energy usage.

Many energy production facilities are in remote or hostile environments such as desert regions for solar arrays, steep hillsides for wind farms, and hazardous facilities for nuclear reactors. Additionally, data may need real-time or near real-time response for critical response to energy production control systems (much like manufacturing systems). This can impact how an IoT system is deployed in this category. We will talk about issues of real-time responsiveness later in this book.

Energy IoT use cases

The following are some of the use cases for energy IoT:

- Oil rig analysis of thousands of sensors and data points for efficiency gains
- Remote solar panel monitoring and maintenance
- Hazardous analysis of nuclear facilities
- Smart electric meters in a citywide deployment to monitor energy usage and demand
- Real-time blade adjustments as a function of weather on remote wind turbines

Smart city

Smart city is a phrase used to imply connecting intelligence to what had been an unconnected world. Smart cities are one of the fastest growing segments, and show substantial cost/benefit ratios especially when we consider tax revenues. Smart cities also touch citizens' lives through safety, security, and ease of use. For example, several cities such as Barcelona are fully connected and monitor trash containers and bins for pickup based on the current capacity, but also the time since the last pickup. This improves the trash collection efficiency allowing the city to use fewer resources and tax revenue in transporting waste, but also eliminates potential smells and odors of rotting organic material. Smart cities are also impacted by government mandates and regulations (as we will explore later), therefore there are ties to the government segment.

One of the characteristics of smart city deployment may be the number of sensors used. For example, a smart camera installation on each street corner in New York would require over 3,000 cameras. In other cases, a city such as Barcelona will deploy nearly one million environmental sensors to monitor electric usage, temperature, ambient conditions, air quality, noise levels, and parking spaces. These all have low bandwidth needs versus a streaming video camera, but the aggregate amount of data transmitted will be nearly the same as the surveillance cameras in New York. These characteristics of quantity and bandwidth need to be considered in building the correct IoT architecture.

Smart city IoT use cases

Some of the smart city IoT use cases are as follows:

- Pollution control and regulatory analysis through environmental sensing
- Microclimate weather predictions using citywide sensor networks
- Efficiency gains and improved costs through waste management service on demand
- Improved traffic flow and fuel economy through smart traffic light control and patterning
- Energy efficiency of city lighting on demand
- Smart snow plowing based on real-time road demand, weather conditions, and nearby plows
- Smart irrigation of parks and public spaces, depending on weather and current usage
- Smart cameras to watch for crime and real-time automated AMBER Alerts
- Smart parking lots to automatically find best space parking on demand
- Bridge, street, and infrastructure wear and usage monitors to improve longevity and service

Government and military

City, state, and federal governments, as well as the military, have a keen interest in IoT deployments. Take California's executive order B-30-15 (<https://www.gov.ca.gov/news.php?id=18938>), which states that by 2030 greenhouse gas emissions affecting global warming will be at levels 40 percent below 1990 levels. To achieve aggressive targets like this, environmental monitors, energy sensing systems, and machine intelligence will need to come into play to alter energy patterns on demand while still keeping the California economy breathing. Other cases include projects like the Internet Battlefield of Things, with the intent of providing efficiencies for friendly, personal, and counter-attacks on enemies. This segment also ties into the smart city category when we consider the monitoring of government infrastructures like highways and bridges.

The government's role in the IoT also comes into play in the form of standardization, frequency spectrum allocation, and regulations. Take, for example, how the frequency space is divided, secured, and portioned to various providers. We will see throughout this text how certain technologies came to be through federal control.

Government and military IoT use cases

Following are some of the government and military IoT use cases:

- Terror threat analysis through IoT device pattern analysis and beacons
- Swarm sensors through drones
- Sensor bombs deployed on the battlefield to form sensor networks to monitor threats
- Government asset tracking systems
- Real-time military personal tracking and location services
- Synthetic sensors to monitor hostile environments
- Water level monitoring to measure dam and flood containment

Summary

Welcome to the world of the IoT. As an architect in this new field, we have to understand what the customer is building, and what the use cases require. IoT systems are not a fire-and-forget type of design. A customer expects several things from jumping on the IoT train.

First, there must be a positive reward. That is dependent on your business, and your customer's intent. From my experience, a 5x gain is the target and has worked well for the introduction of new technologies to pre-existing industries. Second, IoT design is, by nature, a plurality of devices. The value of IoT is not a single device or a single location broadcasting data to a server. It's a set of things broadcasting information and understanding the value the information in aggregate is trying to tell you. Whatever is designed must scale or will scale, therefore that needs attention in upfront design.

We now start exploring the topology of an IoT system as a whole then break down individual components throughout the rest of the book.

Remember, data is the new oil.

2

IoT Architecture and Core IoT Modules

The IoT ecosphere starts with the simplest of sensors located in the remotest corners of the earth, and translates analog physical effects into digital signals (the language of the internet). Data then takes a complex journey through wired and wireless signals, various protocols, natural interference, and electromagnetic collisions, before arriving in the ether of the internet. From there, packetized data will traverse various channels arriving at a cloud or large data center. The strength of IoT is not just one signal from one sensor, but the aggregate of hundreds, thousands, potentially millions of sensors, events, and devices.

This chapter starts with a definition of IoT versus machine-to-machine architectures. It also addresses the architect's role in building a scalable, secure, and enterprise IoT architecture. To do that, an architect must be able to speak to the value the design brings to a customer. The architect must also play multiple engineering and product roles in balancing different design choices.

This book will cover everything from the transformation of physical to digital sensing, power systems and energy storage, to manage billions of devices to near meter, near kilometer, and extreme range communication systems and protocols, to network and information theory, to internet protocols for the IoT, to the role of edge routing and gateways. The book then turns to the application of data through cloud and fog computing, as well as advanced machine learning and complex event processing. The final content investigates security and the vulnerability of the largest attack surface on earth.

IoT ecosystem

These industries will rely on the hardware, software, and services provided by the bulk of the IT industry. Nearly every major technology company is investing or has invested heavily in IoT space. New markets and technologies have already formed (and some have collapsed or been acquired). Throughout this book, we will touch on nearly every segment in information technology, as they all have a role in IoT:

- **Sensors:** Embedded systems, real-time operating systems, energy-harvesting sources, **Micro-Electro-Mechanical Systems (MEMS)**.
- **Sensor communication systems:** Wireless personal area networks reach from 0 cm to 100 m. Low-speed and low-power communication channels, often non-IP based have a place in sensor communication.
- **Local area networks:** Typically, IP-based communication systems such as 802.11 Wi-Fi used for fast radio communication, often in peer-to-peer or star topologies.
- **Aggregators, routers, gateways:** Embedded systems providers, cheapest vendors (processors, DRAM, and storage), module vendors, passive component manufacturers, thin client manufacturers, cellular and wireless radio manufacturers, middleware providers, fog framework providers, edge analytics packages, edge security providers, certificate management systems.
- **WAN:** Cellular network providers, satellite network providers, **Low-Power Wide-Area Network (LPWAN)** providers. Typically using internet transport protocols targeted for IoT and constrained devices like MQTT, CoAP, and even HTTP.
- **Cloud:** Infrastructure as a service provider, platform as a service provider, database manufacturers, streaming and batch processing manufacturers, data analytics packages, software as a service provider, data lake providers, Software-Defined Networking/Software-Defined Perimeter providers, and machine learning services.
- **Data analytics:** As the information propagates to the cloud en-mass. Dealing with volumes data and extracting value is the job of complex event processing, data analytics, and machine learning techniques.
- **Security:** Tying the entire architecture together is security. Security will touch every component from physical sensors to the CPU and digital hardware, to the radio communication systems, to the communication protocols themselves. Each level needs to ensure security, authenticity, and integrity. There cannot be the weak link in a chain, as the IoT will form the largest attack surface on earth.

This ecosystem will need talents from the body of engineering disciplines, such as device physics scientists developing new sensor technologies and many-year batteries. Embedded system engineers working on driving sensors at the edge. Network engineers capable of working in a personal area network or wide area network as well as a Software-Defined Networking. Data scientists working on novel machine learning schemes at the edge and in the cloud. DevOps engineers to successfully deploy scalable cloud solutions as well as *fog* solutions. IoT will also need service vendors such as solution provision firms, system integrators, value-added resellers, and OEMs.

IoT versus machine to machine

One common area of confusion in the IoT world is what separates it from the technologies that defined **machine to machine (M2M)**. Before IoT became part of the mainstream vernacular, M2M was the hype. M2M and IoT are very similar technologies, but there is a significant difference:

- **M2M:** It is a general concept involving an autonomous device communicating directly to another autonomous device. Autonomous refers to the ability of the node to instantiate and communicate information with another node without human intervention. The form of communication is left open to the application. It may very well be the case that an M2M device uses no inherent services or topologies for communication. This leaves out typical internet appliances used regularly for cloud services and storage. An M2M system may communicate over non-IP based channels as well, such as a serial port or custom protocol.
- **IoT:** IoT systems may incorporate some M2M nodes (such as a Bluetooth mesh using non-IP communication), but aggregates data at an edge router or gateway. An edge appliance like a gateway or router serves as the entry point onto the internet. Alternatively, some sensors with more substantial computing power can push the internet networking layers onto the sensor itself. Regardless of where the internet *on-ramp* exists, the fact that it has a method of tying into the internet fabric is what defines IoT.

By moving data onto the internet for sensors, edge processors, and smart devices, the legacy world of cloud services can be applied to the simplest of devices. Before cloud technology and mobile communication became mainstream and cost-effective, simple sensors and embedded computing devices in the field had no good means of communicating data globally in seconds, storing information for perpetuity, and analyzing data to find trends and patterns. As cloud technologies advanced, wireless communication systems became pervasive, new energy devices like lithium-ion became cost-effective, and machine learning models evolved to produce actionable value. This greatly improved the IoT value proposition. Without these technologies coming together when they did, we would still be in an M2M world.

The value of a network and Metcalfe's and Beckstrom's law

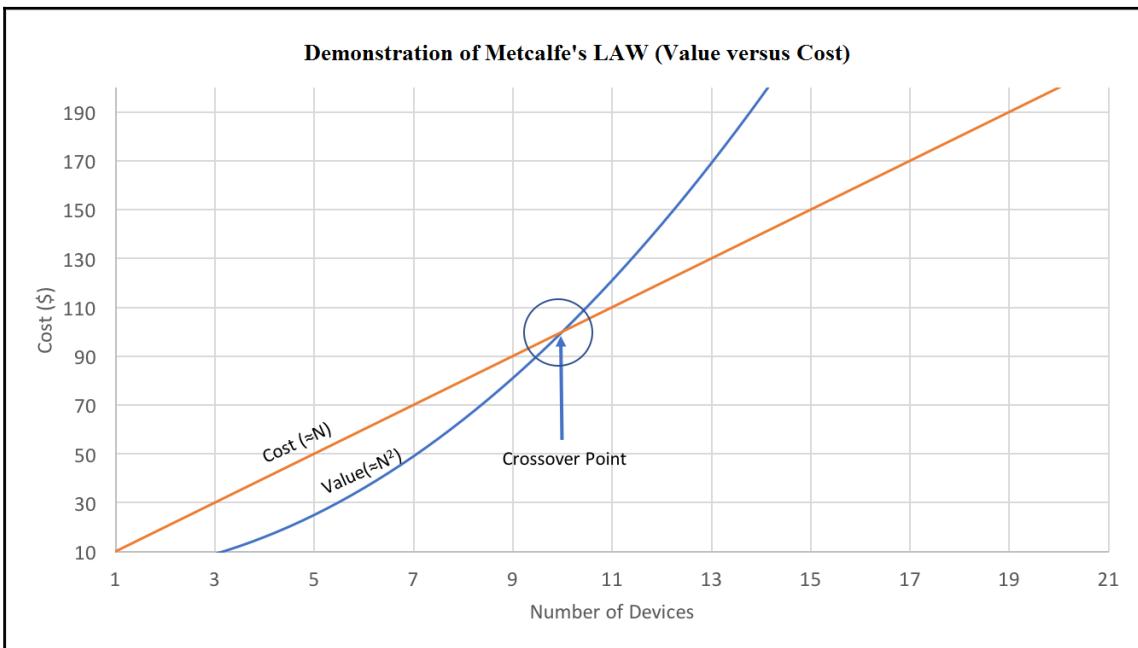
It has been argued that the value of a network is based on Metcalfe's law. Robert Metcalfe in 1980 formulated the concept that the value of any network is proportional to the square of connected *users* of a system. In the case of IoT, users may mean sensors or edge devices. Generally speaking, Metcalfe's law is represented as:

$$V \propto N^2$$

Where:

- V = Value of the network
- N = Number of nodes within the network

A graphical model helps to understand the interpretation as well as the **Crossover Point**, where a positive return on investment (ROI) can be expected:



Metcalfe's Law. Value of a network is represented as proportional to N^2 . The cost of each node is represented as kN where k is an arbitrary constant. In this case, k represents a constant of \$10 per IoT edge sensor. The key takeaway is the crossover point occurs rapidly due to the expansion of value and indicates when this IoT deployment achieves a positive ROI.

A recent example validating Metcalfe's law to the value of blockchains and cryptocurrency trends was recently conducted. We will go much deeper into blockchains in the security chapter.



A recent white paper by Ken Alabi finds that blockchain networks also appear to follow Metcalfe's law, *Electronic Commerce Research and Applications*, Volume 24, C (July 2017), page number 23-29.

Metcalfe's law does not account for service degradation in the event of service degradation as the number of users and/or data consumption grows, but the network bandwidth does not. Metcalfe's law also doesn't account for various levels of network service, unreliable infrastructure (such as 4G LTE in a moving vehicle), or bad actors affecting the network (for example, denial of service attacks).

To account for these circumstances, we use Beckstrom's law:

$$\sum_{i=1}^n V_{i,j} = \sum_{i=1}^n \sum_{k=1}^m \frac{B_{i,j,k} - C_{i,j,k}}{(1 + r_k)^{t_k}}$$

Where:

- $V_{i,j}$: Represents the present value of the network for device i on network j
- i : An individual user or device on the network
- j : The network itself
- k : A single transaction
- $B_{i,j,k}$: The benefit that value k will bring to device i on network j
- $C_{i,j,k}$: The cost of a transaction k to a device i on network j
- r_k : The discount rate of interest to the time of transaction k
- t_k : The elapsed time (in years) to transaction k
- n : The number of individuals
- m : The number of transactions

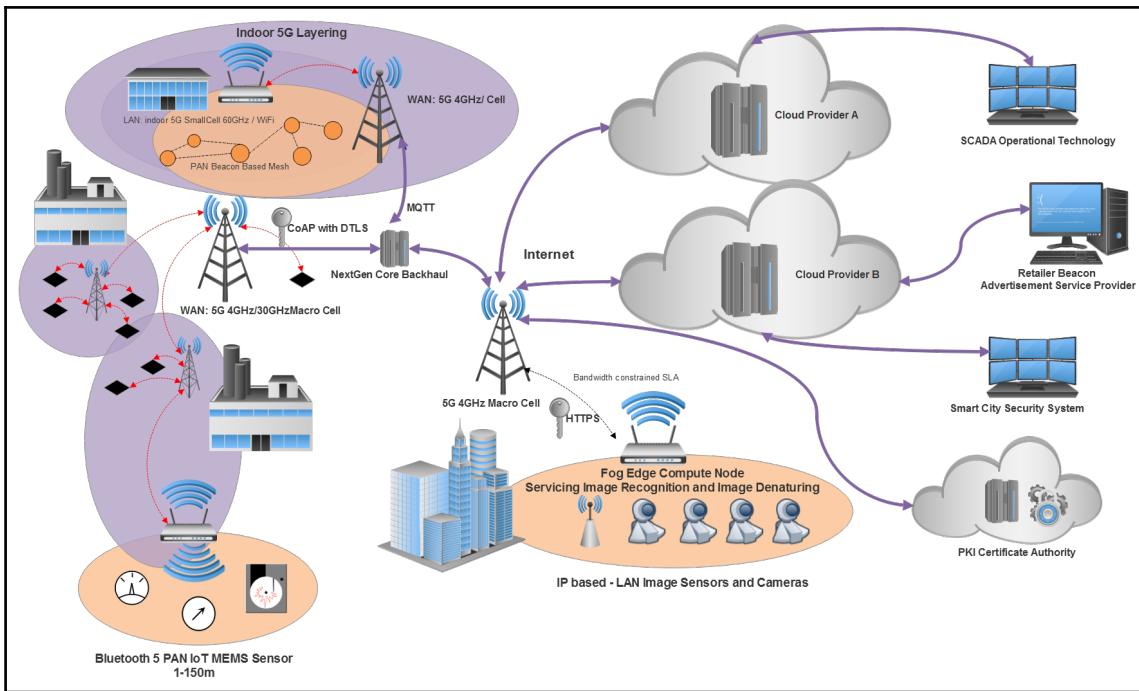
Beckstrom's law teaches us that to account for the value of a network (for example, an IoT solution), we need to account for all transactions from all devices and sum their value. If the network j goes down for whatever reason, what is the cost to the users? This is the impact an IoT network brings and is a more representative real-world attribution of value. The most difficult variable to model in the equation is the benefit of a transaction B . While looking at each IoT sensor, the value may be very small and insignificant (for example, a temperature sensor on some machine is lost for an hour). At other times, it can be extremely significant (for example, a water sensor battery died and a retailer basement is flooded, causing significant inventory damage and insurance adjustments).

An architect's first step in building an IoT solution should be to understand what value they are bringing to what they are designing. In the worst case, an IoT deployment becomes a liability, and actually produces negative value for a customer.

IoT architecture

An IoT architecture as we have eluded to will span many technologies. As an architect, one needs to understand the impact choosing a certain design aspect will have on scalability and other parts of the system. The complexities and relationships of the IoT because significantly more complex than traditional technologies because of the scale but also due to the disparate types of architecture. There is a bewildering number of design choices. For example, as of this writing, there are over 700 IoT service providers alone offering cloud-based storage, SaaS components, IoT management systems, IoT security systems, and every form of data analytics one can imagine. Add to that the number of different PAN, LAN, and WAN protocols that are constantly changing and varying by region. Choosing the wrong PAN protocol could lead to poor communications and significantly low signal quality that can only be resolved by adding more nodes to complete a mesh. The architect needs to consider interference effects in the LAN and WAN—how will the data get off the edge and on the internet? The architect needs to consider resiliency and how costly the loss of data is. Should resiliency be managed within the lower layers of the stack, or in the protocol itself? The architect must also make choices of internet protocols such as MQTT versus CoAP and AMQP, and how that will work if he or she decides to migrate to another cloud vendor. Choices also need consideration with regards to where processing should reside. This opens up the notion of fog computing to process data close to its source to solve latency problems, but more importantly to reduce bandwidth and costs moving data over WANs and clouds. Next, we consider all the choices in analyzing the data collected. Using the wrong analytic engine may result in useless noise or algorithms that are too resource intensive to run on edge nodes. Next, how will queries from the cloud back to the sensor affect the battery life of the sensor device itself? Add to this litany of choice, and we have to layer on security as the IoT deployment we have built is now the largest attack surface in our city. As you can see, the choices are many and have relationships with one another.

There are now over 1.5 million different combinations of architectures to choose from:



IoT Design Choices. The full spectrum of various levels of IoT architecture from the sensor to cloud and back.

Role of an architect

The term **architect** is often used in technical disciplines. There are software architects, system architects, and solution architects. Even within specific domains such as computer science and software engineering, you may see people with the title SaaS architect, cloud architect, data science architect, and so on. These are individuals that are recognized experts with tangible skills and experience in a domain. These types of specialized vertical domains cross a number of horizontal technologies. In this book, we are targeting the IoT architect. This is a horizontal role, meaning it will touch a number of these domains and bring them together for a usable, secure, and scalable system.

We will go as deep as necessary to understand an entire IoT system to bring a system together. At times, we will go into pure theory, such as information and communication theory. Other times, we will brush on topics that are on the periphery of IoT systems, or are rooted in other technologies. By reading and referencing this book, the architect will have a go-to guide on different aspects of IoT that are all needed to build a successful system. Whether you are disciplined in EE or CS, or have domain expertise in cloud architectures, this material will help you understand a holistic system—which should be, by definition, part of the role of an architect.

This book is also intended for global and mass scaling. While every topic can be used for hobbyist and maker movements, this is intended to scale to global enterprise systems on the order of thousands to millions of edge devices.

Part 1 – Sensing and power

The internet starts or ends with an event: a simple motion, a temperature change, perhaps an actuator moving on a lock. Unlike many IT devices in existence, IoT in a large part is about a physical action or event. It responds to affect a real-world attribute. Sometimes this involves considerable data being generated from a single sensor, such as auditory sensing for preventative maintenance of machinery. Other times, it's a single bit of data indicating vital health data from a patient. Whatever the case may be, sensing systems have evolved and made use of Moore's law in scaling to sub-nanometer sizes and significantly reduced costs. When one talks about the *billions* of devices in IoT, this is what they are referring to and why there will be billions of devices in reality. This chapter explores the depths of MEMs, sensing, and other forms of low-cost edge devices from a physical and electrical point of view. The chapter also details the necessary power and energy systems to drive these edge machines. We can't take power for granted at the edge. Collections of billions of small sensors will still require a massive amount of energy in whole to power. We will revisit power throughout this book, and how innocuous changes in the cloud can severely impact the overall power architecture of a system.

Part 2 – Data communication

A significant portion of this book surrounds connectivity and networking. There are countless other sources that dive deep into application development, predictive analytics, and machine learning. This book too will cover those topics, but an equal amount of emphasis is given to data communications. The IoT wouldn't exist without significant technologies to move data from the remotest and most hostile environment to the largest data centers at Google, Amazon, Microsoft, and IBM. The acronym IoT contains the word *internet*, and because of that, we need to dive deep into networking, communications, and even signal theory. The starting point for IoT isn't sensors or the application; it's about connectivity, as we will see throughout this book. A successful architect will understand the constraints of internetworking from a sensor to a WAN and back again.

This communication and networking section starts with theory and mathematical foundations of communication and information. Preliminary tools and models are needed by a successful architect not only to understand why certain protocols are constrained, but also to design future systems that scale successfully at IoT levels. These tools include wireless radio dynamics like range and power analysis, signal-to-noise ratio, path loss, and interference. The chapter also details foundations of information theory and constraints that affect overall capacity and quality of data. The foundations of Shannon's law will be explored. The wireless spectrum is also finite and shared, as an architect deploying a massive IoT system will need to understand how the spectrum is allocated and governed. Theory and models explored in this chapter will be reused in other sections of the book.

Data communication and networking will then build up from the near-range and near-meter communication systems known as **personal area networks (PANs)**, typically using non-internet protocol messages. The chapter on PAN will include the new Bluetooth 5 protocol and mesh, as well as Zigbee and Z-Wave in depth. These represent the plurality of all IoT wireless communication systems. Next, we explore wireless local area networks and IP-based communication systems including the vast array of IEEE 802.11 Wi-Fi systems, Thread, and 6LoWPAN. The chapter also investigates new Wi-Fi standards such as 802.11p for in-vehicle communication.

The section concludes with long-range communication using cellular (4G LTE) standards, and dives deep into the understanding and infrastructure to support 4G LTE and new standards dedicated to IoT and machine to machine communication, such as Cat-1 and Cat-NB. The chapter also details the most promising features of the new 5G standard being ratified and designed to prepare the architect for future long-range transmissions where every device is connected in some capacity. A proprietary protocol like LoRaWAN and Sigfox are also explored to understand the differences between architectures.

Part 3 – Internet routing and protocols

To bridge data from sensors to the internet, two technologies are needed: gateway routers and supporting IP-based protocols designed for efficiency. This section explores the role of router technologies at the edge for bridging sensors on a PAN network to the internet. The role of the router is especially important in securing, managing, and steering data. Edge routers orchestrate and monitor underlying mesh networks and balance and level data quality. The privatization and security of data is also critical. The chapter will explore the router role in creating virtual private networks, virtual LANs, and software-defined wide-area networks. There literally may be thousands of nodes serviced by a single edge router, and in a sense, it serves as an extension to the cloud, as we will see in the [Chapter 10, Cloud and Fog Topologies](#).

The section continues with the protocols used in IoT communication between nodes, routers, and clouds. The IoT has given way to new protocols rather than the legacy HTTP and SNMP type of messaging used for decades. IoT data needs efficient, power-aware, and low-latency protocols that can be easily steered and secured in and out of the cloud. This chapter explores protocols such as the pervasive MQTT, as well as AMQP and CoAP. Examples are given to illustrate their use and efficiency.

Part 4 – Fog and edge compute, analytics, and machine learning

At this point, we must consider what to do with the data streaming in from edge nodes into a cloud service. First, we begin by talking about the aspects of cloud architectures such as SaaS, IaaS, and PaaS systems. An architect needs to understand the data flow and typical design of cloud services (what they are and how they are used). We use OpenStack as a model of cloud design and explore the various components from ingestor engines, to data lakes, to analytics engines. Understanding the constraints of cloud architectures is also important to make a good judgment on how a system will deploy and scale. An architect must also understand how latency can affect an IoT system. Alternatively, not everything belongs in the cloud. There is a measurable cost in moving all IoT data to a cloud versus processing it at the edge (edge processing), or extending cloud services downward into an edge router (fog computing). The section dives deep into new standards of fog compute, such as the OpenFog architecture.

Data that has been transformed from a physical analog event to a digital signal may have actionable consequences. This is where the analytics and rules engines of the IoT come in to play. The level of sophistication for an IoT deployment is dependent on the solution being architected. In some situations, a simple rules engine looking for anomalous temperature extremes can easily be deployed on an edge router monitoring several sensors. In other situations, a massive amount of structured and unstructured data may be streaming in real time to a cloud-based data lake, and require both fast processing for predictive analytics, and long-range forecasting using advanced machine learning models, such as recurrent neural networks in a time-correlated signal analysis package. This chapter details the uses and constraints of analytics from complex event processors, to Bayesian networks, to the inference and training of neural networks.

Part 5 – Threat and security in IoT

We conclude the book with a survey of IoT compromises and attacks. In many cases, IoT systems will not be secured in a home, or in a company. They will be in the public, in very remote areas, in moving vehicles, or even inside a person. The IoT represents the single biggest attack surface for any type of cyber attack. We have seen countless academic hacks, well-organized cyber assaults, and even nation-state security breaches with IoT devices being the target. This chapter will detail several aspects of such breaches and types of remediation any architect must consider to make a consumer or enterprise IoT deployment a *good citizen* of the internet. We explore the proposed Congressional Act to secure the IoT, and understand the motivation and impact of such government mandate.

The chapter will checklist the typical security provisions needed for IoT, or any network component. Details of new technologies such as blockchains and Software Defined Perimeters will also be explored to provide insight into future technologies that will be needed to secure the IoT.

Summary

This book will bridge the spectrum of technologies that comprise the IoT. We summarized the domains and topics covered in this chapter. An architect must be cognizant of the interactions between these disparate engineering disciplines to build a system that is scalable, robust, and optimized. An architect will also be called upon to provide supporting evidence that the IoT system provides a value to the end user or the customer. Here, we learned about the application of Metcalfe's and Beckstrom's laws as tools for supporting an IoT deployment.

In the next chapter and those that follow, we will dive into IoT architecture from the sensor to the cloud, and everything in between.

3

Sensors, Endpoints, and Power Systems

The **Internet of Things (IoT)** begins with sources of data or devices that perform an action. These we call **endpoints**, and they are the things associated with the internet. When one discusses IoT, in general, the actual sources of data are often overlooked. These sources are sensors outputting a stream of time-correlated data that must be transmitted securely, possibly analyzed, and possibly stored. The value of IoT is in the data in aggregate. Therefore, the data a sensor provides is crucial. However, for an architect, it is critical to understand the data as well as how the data is interpreted. Besides understanding what data is collected and how it is acquired, in a massive IoT deployment, it is useful to know what can be sensed, and what the constraints are for various sensors. For example, a system must account for lost devices and erroneous data. An architect must understand the reasons why data may not be reliable from sensors, and how a sensor can fail in the field. Essentially, we are connecting the analog world to the digital. The majority of connected things will be sensors, so it is important to understand their role.

This is what the IoT is in a nutshell. The massive growth of connected things and objects will be sensors and actuators, so it is important to understand their relationship in the architecture. This chapter will highlight sensor devices from an electronics and system point of view. It is important to understand the principles of what is being measured, and why. One should ask, "What type of sensor or edge device should I consider for the problem I'm trying to solve?" An architect should consider aspects of cost, features, size, usable life, and precision when deploying an IoT solution. Additionally, the power and energy of edge devices are rarely addressed in IoT literature, but are critical to building reliable and long-lasting technology. The reader should leave this chapter with a high-level grasp of sensor technology and its constraints.

In this chapter, we will cover the following topics:

- Sensing devices from thermocouples to MEMS sensors to vision systems
- Energy generation systems
- Energy storage systems

Sensing devices

We begin by focusing on the sensing or input devices. These come in a variety of forms and complexities, from simple thermocouples to advanced video systems. When one refers to the "billions of IoT things", this section will represent a broad range of sensing things. One reason IoT is a large growth area is the fact that these sensing systems have been reduced in size and cost substantially, with advances in semiconductor fabrication and micromachining.

Thermocouples and temperature sensing

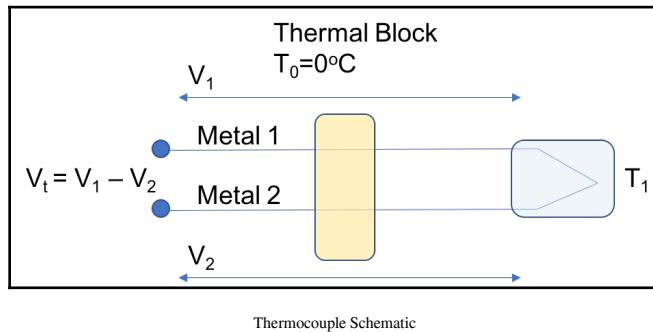
Temperature sensors are the most prevalent form of sensor products. They exist just about everywhere. From smart thermostats to IoT cold storage logistics, from refrigerators to industrial machinery, they are prevalent, and are most likely the first sensing device you will be exposed to in an IoT solution.

Thermocouples

A **thermocouple** (or TC) is a form of temperature-sensing apparatus that doesn't rely on an excitation signal to operate. Therefore, they produce very small signals (often microvolts in amplitude). Two wires of two different materials meet where a temperature measurement is to be sampled. Each metal develops a voltage differential independently of each other. This effect is known as the **Seebeck electromotive effect**, in which the difference between the voltage of the two metals has a nonlinear relationship to the temperature.

The magnitude of the voltage depends on the metal material chosen. It is critical that the ends of the wires are thermally isolated from the system (and the wires need to be at the same controlled temperature). In the following diagram, you will see a thermal block whose temperature is controlled with a sensor. This is usually controlled via a technique called the **cold junction compensation**, where the temperature can vary, but is measured by the block sensor accurately.

When sampling the voltage differential, the software will usually have a look-up table to derive the temperature based on the nonlinear relationship of the metals chosen:

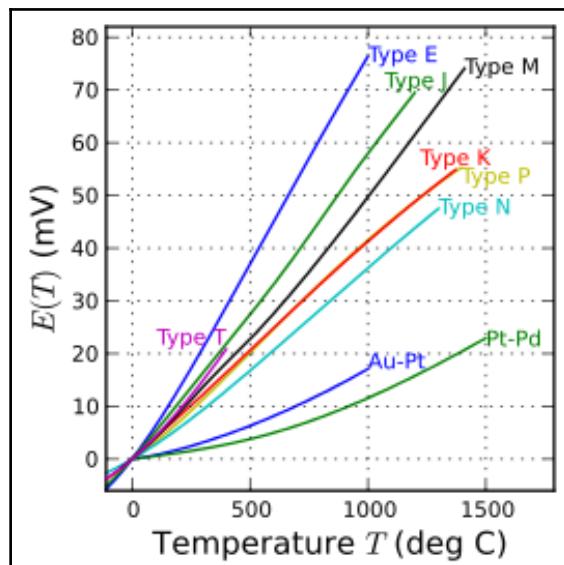


Thermocouple Schematic

Thermocouples should be used for simple measurements. The system can vary in accuracy as well, since subtle impurities can affect the wire composition and cause a mismatch in the lookup tables. Precision-grade thermocouples may be needed, but come at a higher cost. Another effect is aging. Since thermocouples are often used in industrial settings, high-heat environments can degrade the accuracy sensors over time. Therefore, IoT solutions must account for changes over the life of a sensor.

Thermocouples are good for wide temperature ranges, are color-coded for different metal combinations, and labeled by type (for example, E, M, PT-PD, to name a few). In general, these sensors are good for long distance measurements with long leads, and are often used in industrial and high-temperature environments.

Shown below are various thermocouple metal types and their respective energy linearities across a range of temperatures.

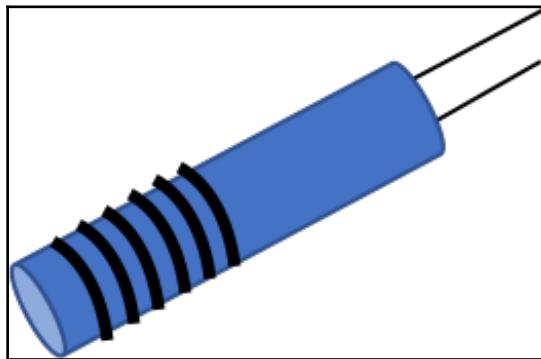


Thermocouple Type Characterization $E(T):T$

Resistance Temperature Detectors

Resistance Temperature Detectors (RTDs) operate within a narrow range of temperatures, but have much better accuracy than thermocouples (below 600 degrees Celsius). These are usually constructed with very fine platinum wire tightly wrapped around ceramic or glass. This produces a resistance-to-temperature relationship. Because it is a resistance-based measurement, an excitation current is needed to operate an RTD (1 mA).

The resistance of an RTD follows a predefined slope. RTDs are specified with a base resistance. A 200 PT100 RTD has a slope of 0.00200 Ohms/degrees Celsius from 0 to 100 degrees Celsius. Within that range (0 to 100 degrees Celsius), the slope will be linear. RTDs come in two, three, and four-wire packages, with four-wire models used for high-accuracy calibration systems. RTDs are often used with bridge circuits to increase resolution, with software linearizing the results:



Wire-wound RTD

RTDs are rarely used above 600 degrees Celsius, which limits their application in industry. At high temperatures, the platinum can become contaminated, leading to incorrect results; however, when measuring within their specified range, RTDs are fairly stable and accurate.

Thermistors

The last temperature sensing device is the **thermistor**. These are also resistance-based relationship sensors like RTDs, but produce a higher degree of change for a given temperature than an RTD. Essentially, these are resistors that vary based on temperature. They are also used in circuits to alleviate in-rush currents. Whereas an RTD has a linear relationship with a change in temperature, thermistors have a highly nonlinear relationship, and are suitable where a high resolution is needed for a narrow temperature range. There are two types of thermistors: NTC, where resistance decreases as temperatures rise; and PTC, where resistance increases with rising temperatures. The main difference from an RTD materially is the use of ceramics or polymers, whereas metals are the basis of RTDs.

Thermistors are found in medical devices, scientific equipment, food handling equipment, incubators, and home appliances such as thermostats.

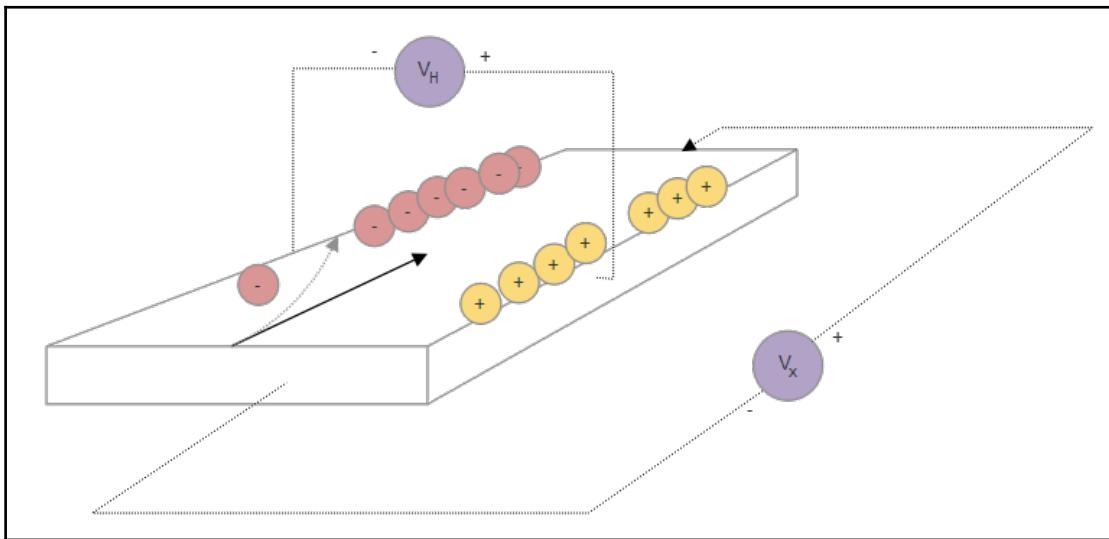
Temperature sensor summary

In summary, the following table highlights the use cases and benefits of particular temperature sensors:

Category	Thermocouples	Resistance Temperature Detectors	Thermistors
Temperature Range (degrees Celsius)	-180 to 2,320	-200 to 500	-90 to 130
Response Time	Fast (microseconds)	Slow (seconds)	Slow (seconds)
Size	Large (~1 mm)	Small (5 mm)	Small (5 mm)
Accuracy	Low	Medium	Very high

Hall effect sensors and current sensors

A hall effect sensor consists of a strip of metal that a current runs across. A stream of charged particles passing through a magnetic field will cause the beam to deflect from a straight line. If a conductor is placed in the magnetic field perpendicular to the electron flow, it will gather charge carriers and produce a voltage differential between the positive side of the metal strip and the negative side. This will produce a voltage differential that can be measured. The differential is called the **Hall voltage**, which is responsible for the phenomenon known as the **Hall effect**. This is illustrated in the following image. If a current is applied to a metal strip (as shown in the diagram) within a magnetic field, electrons will be attracted to one side of the strip, and holes to the other (see the curved line). This will induce an electric field that can be measured. If the field is strong enough, it will cancel out the magnetic force, and electrons will follow the straight line:



Example of the Hall effect.

Current sensors use the Hall effect for measuring the AC and DC currents of a system. There are two forms of current sensors: open loop, and closed loop. Closed loops are more expensive than open loop sensors, and are often used in battery-powered circuits.

Typical uses of Hall sensors include position sensing, magnetometers, highly reliable switches, and water level detection. They are used in industrial sensors to measure the rotational velocity of different machines and motors. Additionally, these devices can be created very inexpensively, and can tolerate harsh environmental conditions.

Photoelectric sensors

The detection of light and light intensity is used in many IoT sensor devices, such as security systems, smart switches, and smart street lighting. As the name implies, a photoresistor varies in resistance depending on light intensity, while photodiodes convert light into an electrical current.

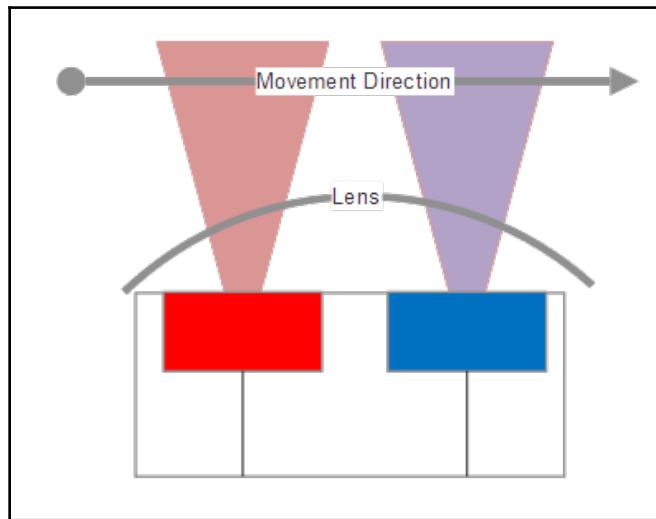
Photoresistors are manufactured using a high-resistance semiconductor. The resistance decreases as more light is absorbed. In the dark, a photoresistor can have quite a high resistance (in the megaohm range). Photons absorbed by the semiconductor allow electrons to jump to the conduction band and conduct electricity. Photoresistors are wavelength sensitive, depending on their type and manufacturer; however, photodiodes are true semiconductors with a p-n junction. The device responds to light by creating an electron-hole pair. The holes move towards an anode, the electrons migrate to a cathode, and a current is produced. Traditional solar cells operate in this photovoltaic mode, producing electricity. Alternatively, a reverse bias can be used on the cathode to improve the latency and response time if needed:

Category	Photoresistor	Photodiode
Light Sensitivity	Low	High
Active/Passive (Semiconductor)	Passive	Active
Temperature Sensitivity	Highly sensitive	Low
Latency to Light Changes	Long (10 milliseconds on, 1 second off)	Short

PIR sensors

Pyroelectric Infrared (PIR) sensors contain two slots filled with material that reacts to IR radiation and heat. Typical use cases are security, or warm-body movement. In its simplest form, a Fresnel lens sits on top of the PIR sensor, allowing both slots to form a widening arc outward. These two arcs create detection zones. When a warm body enters one of the arcs, or leaves one, it generates a signal that is sampled. The PIR sensors use a crystalline material that generates current when subjected to IR radiation. A **Field Effect Transistor (FET)** detects the change in current, and sends the signal to an amplification unit. PIR sensors respond well in the 8 to 14 um range, which is typical of a human body.

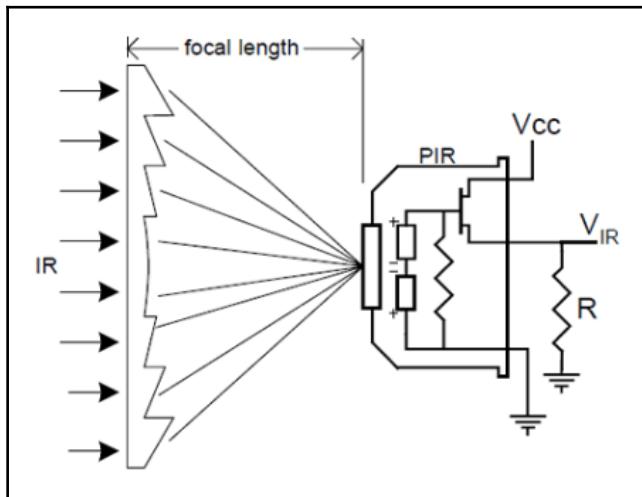
The following diagram illustrates two IR regions detecting two zones. While this is fine for some purposes, generally we need to inspect an entire room or area for movement or activity:



PIR Sensor. Two elements responding to a source of IR moving across its field of view.

To scan a larger area with a single sensor requires multiple Fresnel lenses that condense light from regions of the room to create distinct regions on the PIR array. This also has the effect of condensing the infrared energy to discrete FET areas. Typically, such devices allow the architect to control the sensitivity (range) as well as the hold time.

The hold time specifies for how long to output a motion event after an object has been detected moving across the path of the PIR. The shorter the hold time, the more events may be output. Below is a diagram of a typical PIR sensor with a Fresnel lens focusing on the substrate by a fixed focal length:



Left: Fresnel Lens condensing IR region onto PIR sensor. Cypress Microsystems Application Note AN2105.

LiDAR and active sensing systems

This section will cover active sensing systems. We have discussed many passive sensors that simply respond to environmental changes. Active sensing involves broadcasting a signal that is referenced to measure the environment either spatially or sensorily. While this area is broad, we will concentrate on LiDAR as a basis for active sensing systems.

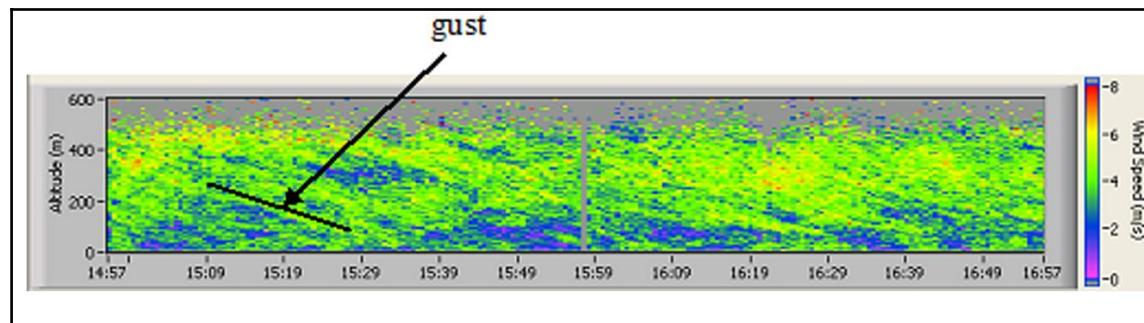
LiDAR stands for Light Detecting and Ranging. This type of sensor measures the distance to a target by measuring a laser pulse reflection on the target. Where a PIR sensor will detect movement within a range, LiDAR is capable of measuring a range. The process was first demonstrated in the 1960s, and is now used pervasively in agriculture, automated and self-driving vehicles, robotics, surveillance, and environmental studies. This type of active sensing machine is also capable of analyzing anything that crosses its path. They are used to analyze gases, atmospheres, cloud formations and compositions, particulates, the speed of moving objects, and so on.

LiDAR is an active sensor technology, and broadcasts laser energy. As a laser hits an object, some of the energy will be reflected back towards the LiDAR emitter. The lasers used are typically in the 600 to 1000 nm wavelength, and are relatively inexpensive. Power is constrained for safety reasons to prevent eye damage. Some LiDAR units operate in the 1550 nm range, as that wavelength cannot be focused by the eye, rendering them harmless even at high energy. LiDAR systems are capable of very long ranging and scanning, even from satellites. The laser will pulse up to 150,000 pulses per second, which can reflect off an object back to a photodiode array. The laser apparatus can also sweep the scene via a rotating mirror to build a comprehensive 3D image of the environment. Each beam that is broadcast represents an angle, **Time of Flight (TOF)** measurement, and a GPS location. This allows the beams to form a representative scene.

To calculate the distance to an object, the equation is relatively simple:

$$\text{Distance} = \frac{(\text{Speed of Light} \times \text{Time of Flight})}{2}$$

LiDAR and other active sensors behave in a similar manner. Each has a representative broadcasting signal that returns to a sensor to build an image, or indicate that an event has occurred. These sensors are much more complex than simple passive sensors, and also take more power, cost, and area:



LiDAR: An example of a LiDAR image used to analyze atmospheric wind gusts to protect wind turbines. Image courtesy of NASA.

MEMS sensors

Micro-electromechanical systems (MEMS) have been in the industry since they were first produced in the 1980s; however, the origin of the first MEMS pressure sensor dates back to the 1960s at Kulite Semiconductor, who developed a piezoresistive pressure sensor. Essentially, they incorporate miniaturized mechanical structures that interact with electronic controls. Typically, these sensors are in the 1 to 100 um geometry range. Unlike the other sensors mentioned in this chapter, MEMS mechanical structures can spin, stretch, bend, move, or alter form, which in turn affects an electrical signal. This is the signal captured and measured by one particular sensor.

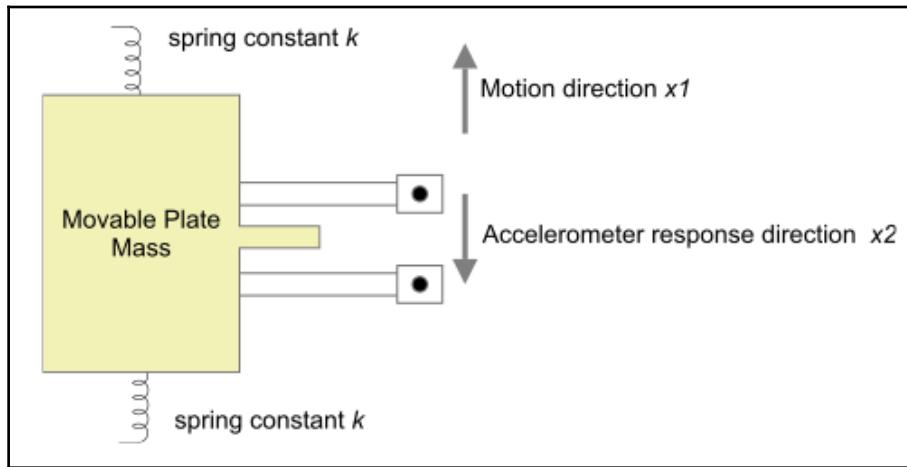
MEMS devices are manufactured in a typical silicon fabrication process using multiple masks, lithography, deposition, and etching processes. MEMS silicon dies are then packaged with other components such as operational amplifiers, analog to digital converters, and support circuitry. Typically, MEMS devices will be fabricated in the relatively large 1 to 100-micron range, whereas typical silicon structures are fabricated at 28 nm or below. The process involves a thin layer deposition and etching to create the 3D structures for a MEMS device.

Besides sensor systems, MEMS devices can be found in the heads of inkjet printers and modern overhead projectors such as **digital light processor (DLP)** projectors. The ability to synthesize MEMS sensing devices into packages as small as a pinhead has and will allow IoT growth into billions of connected things.

MEMS accelerometers and gyroscopes

Accelerometers and gyroscopes are common in many mobile devices today, and are used in positioning and movement tracking, such as with pedometers and fitness trackers. These devices will use a MEMS piezoelectric to produce a voltage in response to movement. Gyroscopes detect rotational motion, and accelerometers respond to changes in linear motion. The following diagram illustrates the basic principle of an accelerometer. Typically, a central mass that is fixed to a calibrated location via a spring will respond to changes in acceleration that are measured by varying capacitance in a MEMS circuit. The central mass will appear to be stationary in response to acceleration in one direction.

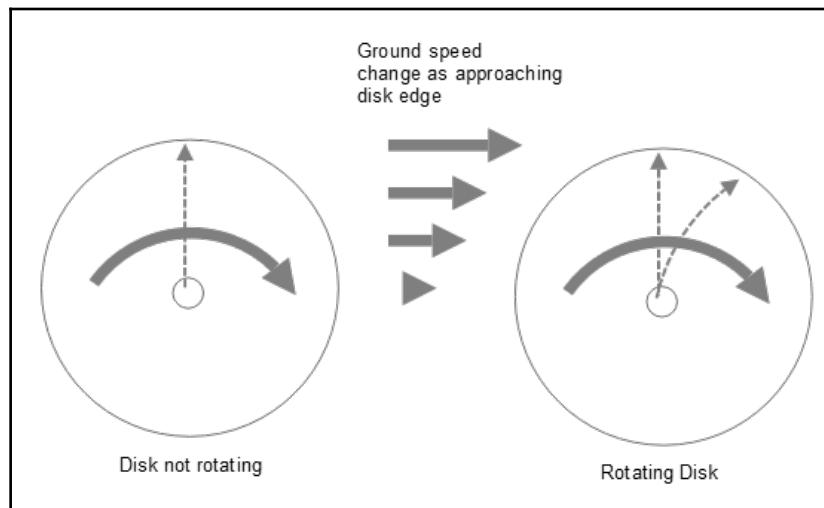
An accelerometer will be synthesized to respond to multiple dimensions (X , Y , Z) rather than one dimension, as shown in this illustration:



Accelerometer: The principle of acceleration measurement using a central mass suspended by a spring. Typically, these will be used in multiple dimensions.

Gyroscopes operate slightly differently. Rather than relying on the motion response to a central mass, gyros rely on the Coriolis effect of a rotating reference frame. The following figure demonstrates the concept. Without increasing velocity, the object would move in an arc, and not reach the northbound target. Moving towards the outer edge of the disk requires additional acceleration to maintain a northbound course.

This is the Coriolis acceleration. In a MEMS device, there is no spinning disk; rather, there is a resonant frequency applied to a series of MEMS-fabricated rings on a silicon substrate:

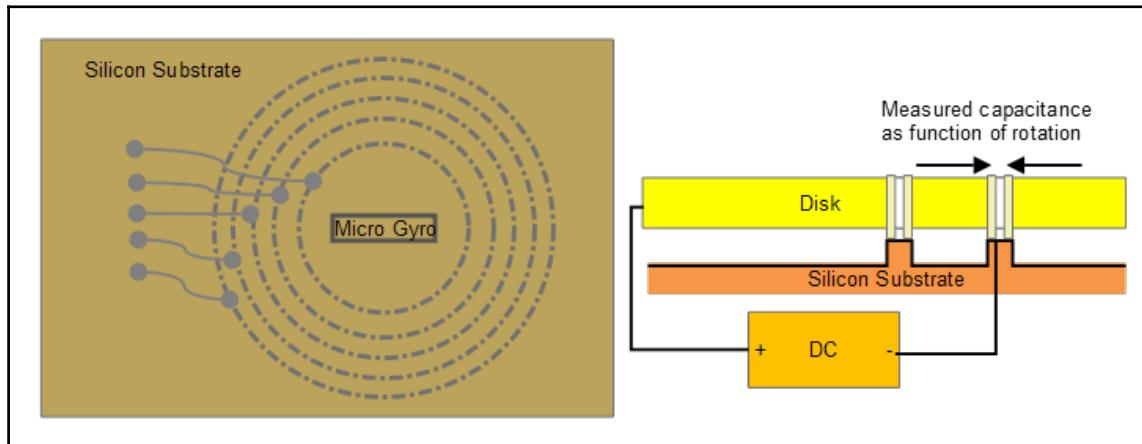


Accelerometer: Effect of a rotating disk on a path moving northward.

The rings are concentric, and cut into small arcs. Concentric rings allow for more area to gauge the accuracy of rotational movement. Single rings would require rigid support beams, and are not as reliable. By bifurcating the rings into arcs, the structure loses rigidity, and is more sensitive to rotational forces. The DC source creates an electrostatic force that resonates within the ring, while the electrodes attached to the rings detect the changes in the capacitor. If the resonating rings are perturbed, the Coriolis acceleration is detected. The Coriolis acceleration is defined by the following equation:

$$\mathbf{a} = -2\omega \times \mathbf{v}$$

This equation states that the acceleration is a product of the rotation of the system and the velocity of the rotating disk, as shown in the preceding diagram, or the resonant frequency of the MEMS device, as shown in the following diagram. Given a DC power source, a force changes the gap size and the overall capacitance of the circuit. The outer electrodes detect deflection in the ring, while inner electrodes provide capacitance measurements:



Right: Concentric cut rings representing the gyro sensor sitting upon a silicon substrate. Left: the disk gap connected to a corresponding.

Both gyroscopes and accelerometers will require power supplies and an op-amp for signal conditioning. After conditioning, the output is ready to be sampled by a digital signal processor.

These devices can be synthesized in very small packages, such as the Invensense MPU-6050, which includes a 6-axis gyro and accelerometer in a small 4 mm x 4 mm x 1 mm package. The device draws on 3.9 mA of current, and is good for low power sensing.

MEMS microphones

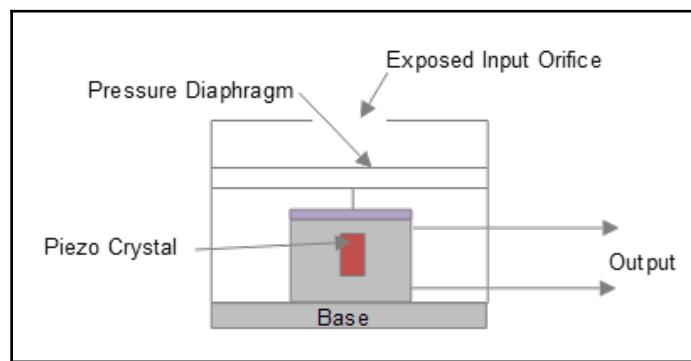
MEMS devices can also be used for sound and vibration detection. These types of MEMS devices are related to the accelerometers previously covered. For IoT deployments, sound and vibration measurements are common in the industrial IoT and predictive maintenance applications. For example, an industrial machine that rotates or mixes a load of material in chemical manufacturing, or in centrifuges to separate mixtures, needs precise leveling. A MEMS sound or vibration unit will typically be used to monitor the health and safety of such equipment.

This type of sensor will require an analog-to-digital converter of sufficient sampling frequency. Additionally, an amplifier is used to strengthen the signal. The impedance of a MEMS microphone is in the order of several hundred ohms (which requires careful attention to the amplifier used). A MEMS microphone can be analog or digital. An analog variety will be biased to some DC voltage, and will be attached to a codec for analog-to-digital conversion. A digital microphone has the ADC close to the microphone source. This is useful when there is signal interference from cellular or Wi-Fi signals near the codec.

The output of a digital MEMS microphone can either be **pulse density modulated (PDM)**, or sent over I^2S format. PDM is a high sampling rate protocol that has the ability to sample from two microphone channels. It does this by sharing a clock and data line, and sampling from one of the two microphones on different clock periods. I^2S doesn't have a high sampling rate, and the decimation at audio rates (Hz to kHz range) results in fair quality. This still allows multiple microphones to be used in sampling, but may not require an ADC at all, since the decimation is happening in the microphone. A PDM, with its high sampling rate, will need to be decimated by a digital signal processor, or DSP.

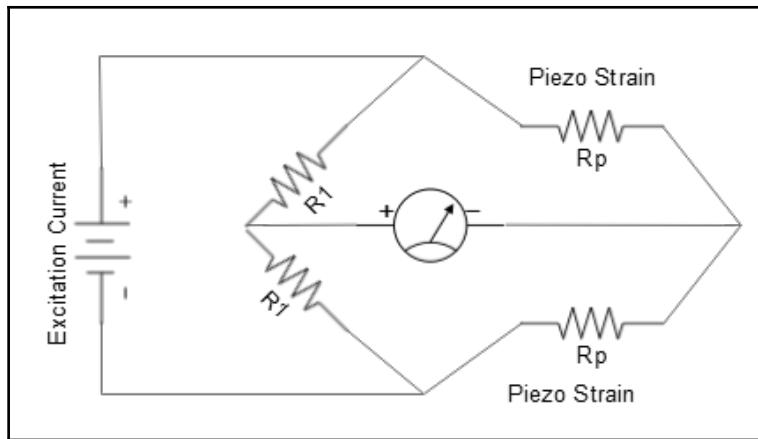
MEMS pressure sensors

Pressure and strain gauges are used in a variety of IoT deployments, from smart cities monitoring infrastructure to industrial manufacturing. These are typically used to measure fluid and gas pressures. The heart of the sensor is a piezoelectric circuit. A diaphragm will be placed above or below a cavity on the piezoelectric substrate. The substrate is flexible, and allows the piezo crystals to change shape. This change in shape results in a directly correlated resistance change in the material:



PressureSensor Anatomy

This type of sensor, as well as others listed in this chapter based on an excitation current, relies on a **Wheatstone bridge** to measure changes. Wheatstone bridges may come in two, four, or six-wire combinations. The change in voltage is measured across the bridge as the piezoelectric substrate flexes and changes resistance:



Wheatstone Bridge used for amplification of MEMS pressure sensor

Smart IoT endpoints

So far, we have examined very simple sensors that simply return information in a binary or analog form that must be sampled. There are, however, IoT devices and sensors that have substantial processing power and performance for the tasks they undertake. Smart sensors include devices such as video cameras and vision systems. Smart sensors can include substantial amounts of processing in the form of high-end processors, digital signal processors, FPGAs, and custom ASICs. In this section, we will explore the details of one form of smart sensor: a vision system.

Vision system

In contrast with the simple sensors explored earlier, vision systems are much more complex, which results in substantial hardware, optics, and imaging silicon. Vision systems start with a lens that observes a scene. A lens provides focus, but also provides more light saturation to the sensing element. In modern vision systems, one of two types of sensing elements is used: **charge-coupled devices (CCD)**, or complementary **metal-oxide (CMOS)** devices. The difference between CMOS and CCD can be generalized as:

- **CCD:** Charge is transported from the sensor to the edge of the chip to be sampled sequentially via an analog-to-digital converter. CCDs create high-resolution and low-noise images. They consume considerable power (100x that of CMOS). They also require a unique manufacturing process.
- **CMOS:** Individual pixels contain transistors to sample the charge and allow each pixel to be read individually. CMOS is more susceptible to noise, but uses little power.

Most sensors are built using CMOS in today's market. A CMOS sensor is integrated into a silicon die that appears as a two-dimensional array of transistors arranged in rows and columns over a silicon substrate. A series of microlenses will sit upon each red, green, or blue sensor focusing incidental rays onto transistor elements. Each of these microlenses attenuates a specific color to a specific set of photodiodes (R, G, or B) that respond to the level of light; however, lenses are not perfect. They can add chromatic aberrations where different wavelengths refract at different rates, which leads to different focal lengths and blur. A lens can also distort an image, causing pincushioning effects.

Next will come a series of steps to filter, normalize, and convert the image several times into a usable digital image. This is the heart of the **image signal processor (ISP)**, and the steps may be followed in this order:

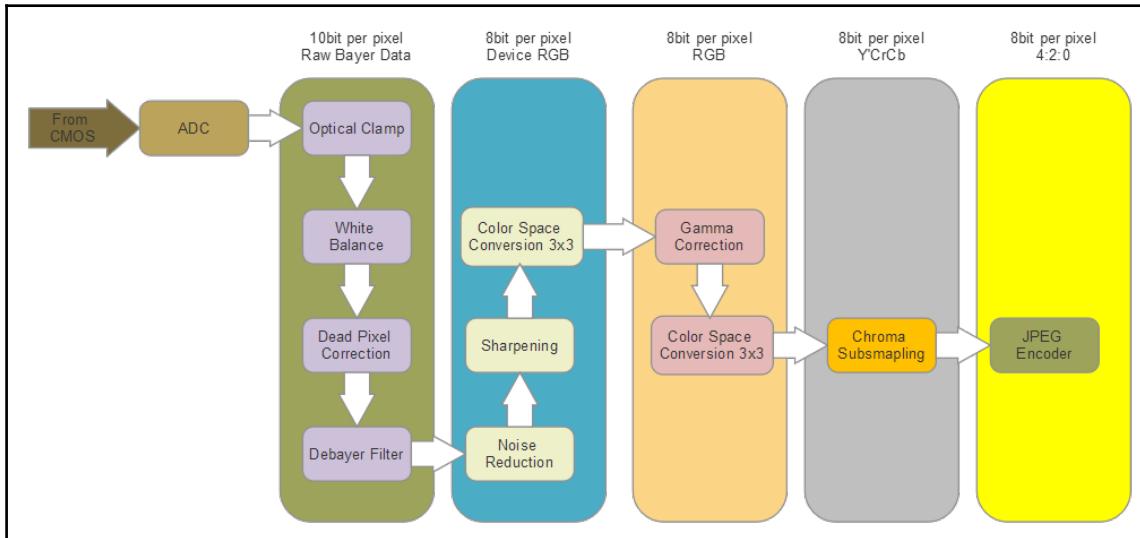


Image Sensor: Typical image signal processor pipeline for color video.

Note the numerous conversions and processing of each stage of the pipeline for each pixel in the image. The amount of data and processing requires substantial custom silicon or digital signal processors. The following lists the functional block responsibilities in the pipeline:

- **Analog-to-digital conversion:** Amplification of sensor signal then converted to digital form (10-bit). Data is read in from the photodiode sensor array as a flattened series of rows/columns representing the image just captured.
- **Optical clamp:** Removes sensor biasing effects due to sensor black level.
- **White balance:** Mimics chromatic display in the eye for different color temperatures, and neutral tones appear neutral. Performed using matrix conversions.
- **Dead pixel correction:** Identifies dead pixels, and compensates for their loss using interpolation, dead pixels are replaced with the average of neighboring pixels.

- **Debayer filtering and demosaicing:** Arranges RGB data to saturate green over red and blue content for luminance sensitivity adjustment. Also creates a planar format of images from sensor-interlaced content. More advanced algorithms preserve edges in images.
- **Noise reduction:** All sensors introduce noise. Noise can be associated with the non-uniformity of pixel sensitivity at a transistor level, or leakage of the photodiode, revealing dark regions. Other forms of noise exist as well. This phase removes white and coherent noise introduced in the image capture through a median filter (3×3 array) across all pixels. Alternatively, a despeckle filter can be used, requiring pixels to be sorted, and other methods exist as well. However, they all walk the pixel matrix.
- **Sharpening:** Applies a blur to an image using a matrix multiplication, then combines the blur with detail in content regions to create a sharpening effect.
- **Color space conversion 3×3 :** Color space conversion to RGB data for RGB particular treatments.
- **Gamma Correction:** Corrects for CMOS image sensor nonlinear response on RGB data to different irradiance. Gamma correction uses a **look-up table (LUT)** to interpolate and correct an image.
- **Color Space Conversion 3×3 :** Additional color space conversion from RGB to Y'CbCr format. YCC was chosen, since Y can be stored at a higher resolution than CbCr without the loss of visual quality. 4:2:2 bit representation.
- **Chroma subsampling:** Due to nonlinearities in RGB tones, this phase corrects images to mimic other media such as film for tonal matching and quality.
- **JPEG encoder:** Standard JPEG compression algorithm.

It should be emphasized here that this is a good example of how complex a sensor can be, and how much data, hardware, and complexity can be attributed to a simple vision system. The amount of data passing through a vision system or camera at a conservative 60 frames per second at 1080p resolution is massive. Assuming all the phases (except JPEG compression) move through an ISP in fixed-function silicon (as in an ASIC) one cycle at a time, the total amount of data processed is 1.368 GB/sec. Accounting for JPEG compression as the last step brings the amount of data to well over 2GB/sec in processing through custom silicon and CPU/DSP cores. One would never stream raw Bayer image video to the cloud for processing - this work must be performed as close to the video sensor as possible.

Sensor fusion

One aspect that needs consideration with all the sensor devices described in this chapter is the concept of **sensor fusion**. Sensor fusion is the process of combining several different kinds of sensor data to reveal more about context than a single sensor can provide. This is important in the IoT space, since a single thermal sensor has no notion of what causes a rapid temperature change. However, when combined with data from other sensors nearby that look at PIR motion detection and light intensity, an IoT system could discern that a large number of people are congregating in a certain area while the sun is shining, and could then make the decision to increase air circulation in a smart building. A simple thermal sensor only records the current temperature value, and has no contextual awareness that the heat is rising due to people congregating and sunlight shining.

With time-correlated data from multiple sensors (edge and cloud), processing can make better decisions based on more data. This is one of the reasons that there will be a large influx of data from sensors to the cloud, and this is causing big data growth. As sensors become cheaper and easier to integrate, as with the TI SensorTag, we will see more combined sensing to provide contextual awareness.

There are two modes of sensor fusion:

- **Centralized:** Where raw data is streamed and aggregated to a central service and fusion occurs there (cloud-based, for example)
- **De-centralized:** Where data is correlated at the sensor (or close to it)

The basis of correlating sensor data is usually expressed through the central limit theorem, where two sensor measurements, x_1 and x_2 , are combined to reveal a correlated measurement, x_3 , based on the combined variances. This is simply adding two measures, and weighting the sum by the variances:

$$x_3 = (\sigma_1^{-2} + \sigma_2^{-2})^{-1}(\sigma_1^{-2}x_1 + \sigma_2^{-2}x_2)$$

Other sensor fusion methods used are Kalman filters and Bayesian networks.

Input devices

There are many other forms of sensing devices we haven't talked about in this chapter, including various gas sensors, humidity sensors, radon and radiation sensors, smoke sensors, ultrasonic sensors, and so on. This chapter, however, should give the reader a working knowledge of the sensor input fundamentals and challenges in choosing the correct sensing options.

Up to this point, we have discussed endpoint devices such as sensors. These devices send a constant stream of data to an edge device, or the cloud. The IoT consists of bi-directional systems. Input may arrive at an endpoint from the cloud, or data may be sent from an endpoint to multiple subscribers in the cloud. This short section covers basic actuators and output devices.

Output devices

Output devices in the IoT ecosystem can be just about anything, from a simple LED to a full video system. Other types of output include actuators, stepper motors, speakers and audio systems, industrial valves, and so on. It stands to reason that these devices need various control systems of differing complexity. Depending on the type of output and the use case they serve, it also should be expected that much of the control and processing needs to be located at the edge, or close to the device (versus complete control in the cloud). A video system, for example, can stream data from cloud providers, but needs output hardware and buffering capabilities at the edge.

In general, output systems can require substantial energy to convert to mechanical movement, thermal energy, or even light. A small hobbyist solenoid to control fluid or gas flow can require 9 to 24 VDC and draw 100 mA to operate reliably and produce five Newton's of force. Industrial solenoids operate in hundreds of volts.

Functional examples (putting it all together)

A collection of sensors is fairly useless unless the data they collect can be transmitted and processed. Whether that is a local embedded controller or sent upstream to a cloud, more hardware is needed to build the system. Typically, the sensors will use established IO interface and communication systems, such as I²C, SPI, UART, SPI, or other low-speed IOs. Other devices such as video systems will need much faster IO to maintain high resolution and fast video frame rates such as MIPI, USB, or even PCI-Express. To communicate wirelessly, the sensors will need to be used with wireless transport hardware like Bluetooth, Zigbee, or 802.11. All this requires extra componentry, which we will cover in this section.

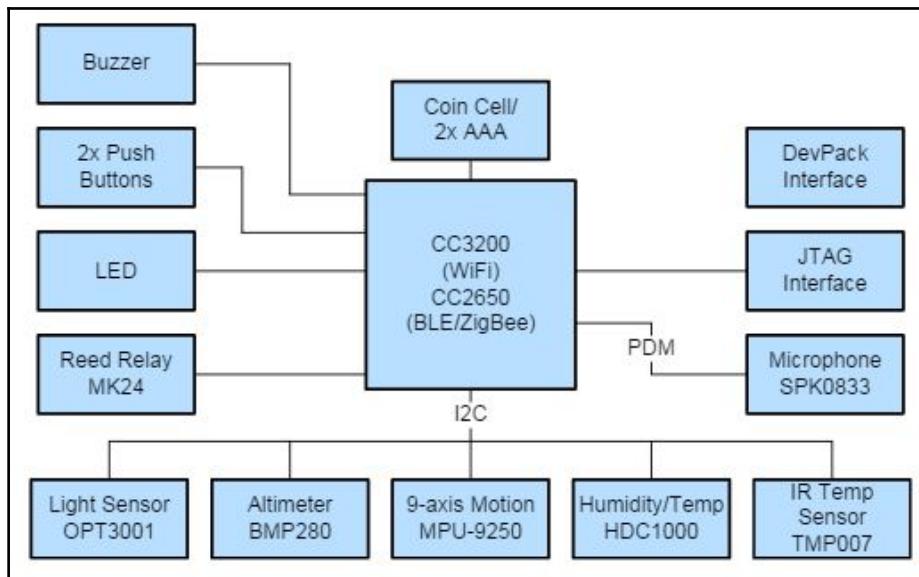
Functional example – TI SensorTag CC2650

The Texas Instruments CC2650 SensorTag is a good example of an IoT sensor module for development, prototyping, and design. SensorTag has the following features and sensors in the package:

- Sensor input
 - Ambient light sensor (TI Light Sensor OPT3001)
 - Infrared temperature sensor (TI Thermopile infrared TMP007)
 - Ambient temperature sensor (TI light sensor OPT3001)
 - Accelerometer (Invensense MPU-9250)
 - Gyroscope (Invensense MPU-9250)
 - Magnetometer (Bosch SensorTec BMP280)
 - Altimeter/Pressure sensor (Bosch SensorTec BMP280)
 - Humidity sensor (TI HDC1000)
 - MEMS microphone (Knowles SPH0641LU4H)
 - Magnetic sensor (Bosch SensorTec BMP280)
 - 2 Push-button GPIOs
 - Reed relay (Meder MK24)

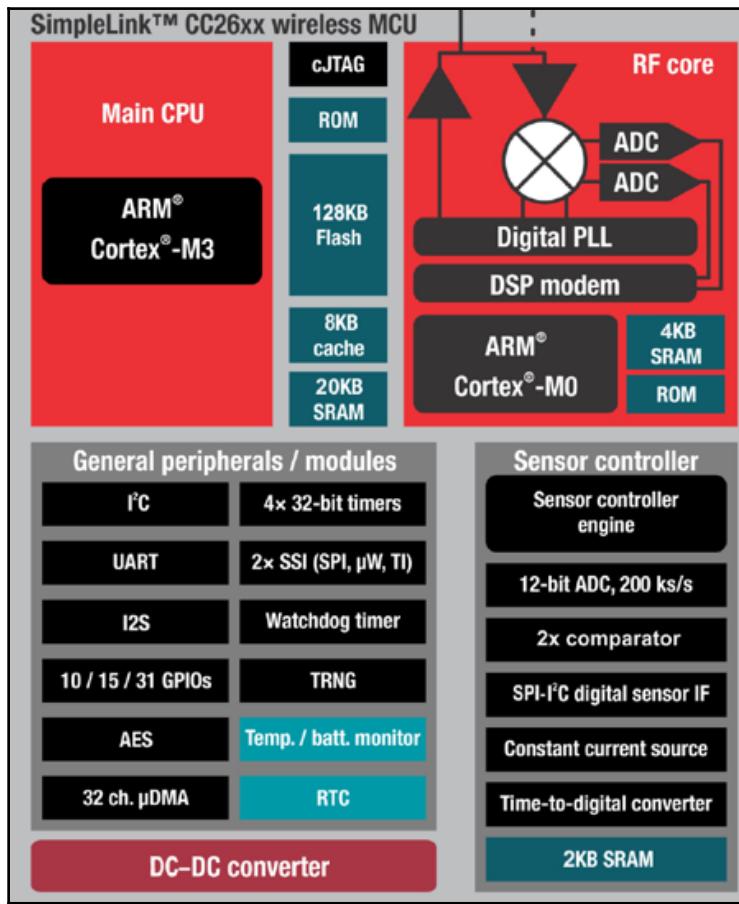
- Output devices
 - Buzzer/speaker
 - 2 LEDs
- Communications
 - Bluetooth Low Energy (Bluetooth Smart)
 - Zigbee
 - 6LoWPAN

This package is powered by a single CR2032 coin cell battery. Finally, the device can be placed into beacon mode (iBeacon), and used as a message broadcaster. Following is the block diagram of the CC2650 SensorTag module:



TI CC2650 SensorTag. Courtesy of Texas Instruments, TI Multi-Standard CC2650 SensorTag Design Guide. Texas Instruments Incorporated, 2015.

The next image is of the block diagram of the MCU. The MCU provides the IO and processing ability using an ARM Cortex M4, and connects over various bus interfaces to sensor components on the module:

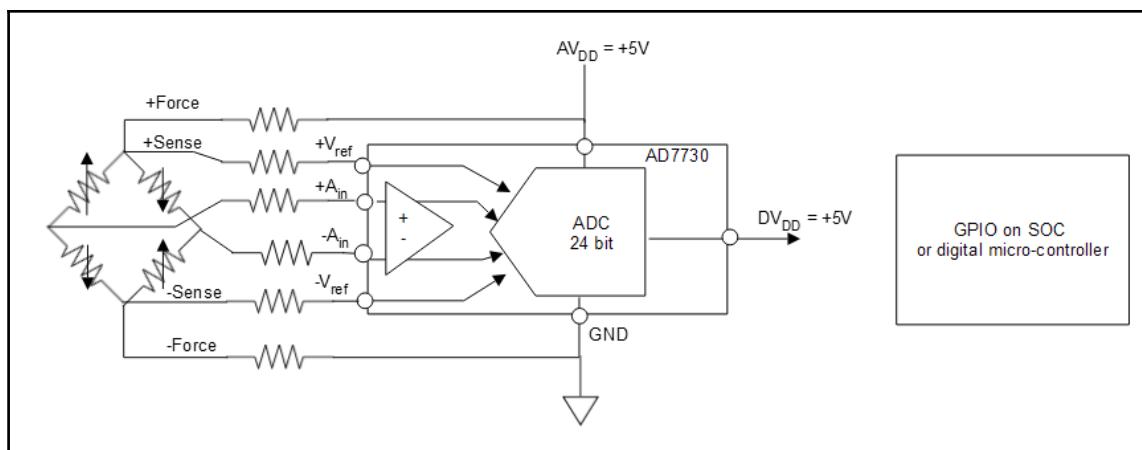


TI CC2650 MCU Block Diagram. Courtesy of Texas Instruments, TI Multi-Standard CC2650 SensorTag Design Guide. Texas Instruments Incorporated, 2015.

This device is packed with a number of sensors, communication systems, and interfaces, but the amount of processing power is limited. The device uses a processing module from TI (MCU CC265), which includes a small ARM Cortex M3 CPU with only 128 KB of flash memory and 20 KB of SRAM. This was chosen for its extremely low power consumption. While power-efficient, this limits the amount of processing and resources on this system. Typically, components like these will need to be accompanied by a gateway, router, cell phone, or some other smart device. Sensor devices such as these that are built for low power and low cost would not have the resources for more demanding applications such as MQTT protocol stacks, data aggregation, cellular communication, or analytics. For that matter, most endpoint sensing devices one will see in the field are simpler than this component, to further reduce cost and power.

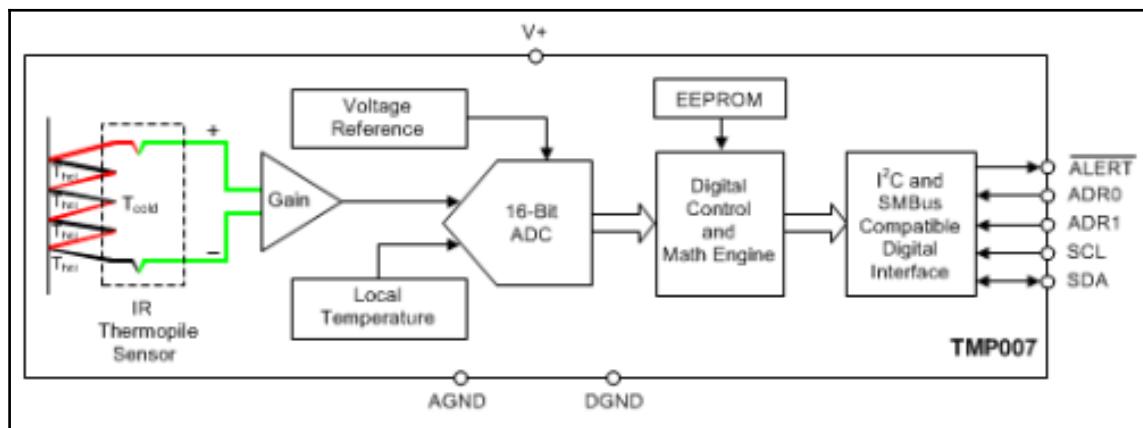
Sensor to controller

In many of the preceding examples for sensing components, the signal will require amplification, filtering, and calibration before going anywhere. Typically, the hardware will need an analog-to-digital converter of some resolution. The following is a simple 24-bit ADC that outputs a 5V signal:



Wheatstone Bridge: connected to AD7730 analog to digital converter as input to a microcontroller or system on a chip

The output can then be raw pulse-modulated data, or a serial interface such as I²C, SPI, or UART to a microcontroller or digital signal processor. A good example of this in a real system is the Texas Instruments infrared thermopile sensor (TMP007). This is a contactless MEMS temperature sensor that absorbs infrared wavelengths and converts them to a reference voltage while using a cold junction reference temperature. It is rated for accurately detecting temperatures in environments between -40 degrees Celsius to +125 degrees Celsius. We can see the components covered in this chapter for this part in the following figure:



TI TMP007 Courtesy of Texas Instruments. TI Multi-Standard CC2650 SensorTag Design Guide. Texas Instruments Incorporated, 2015.

Energy sources and power management

Powering sensors and edge devices reveal a significant problem. When we consider that sensors and edge devices number in the billions, and the fact they will be used in very remote areas, power becomes a challenge. Furthermore, there will be IoT deployments where sensors will be buried undersea, or embedded into concrete infrastructures, further complicating power sources. In this section, we will explore the concepts of energy harvesting and power management. Both are very important concepts in the overall IoT story.

Power management

Managing power is a very broad topic, and spans software and hardware. It is important to understand the role of power management in a successful IoT deployment, and how to manage power efficiently for remote devices and long-lived devices. The architect must build a power budget for the edge device, which includes:

- Active sensor power
- Frequency of data collection
- Wireless radio communication strength and power
- Frequency of communication
- Microprocessor or microcontroller power as a function of core frequency
- Passive component power
- Energy loss from leakage or power supply inefficiency
- Power reserve for actuators and motors

The budget simply reflects the sum of these power contributors subtracted from the source of power (battery). Batteries also do not have a linear power behavior over time. As the battery loses energy capacity while discharging, the amount of voltage will drop curvilinearly. This poses problems for wireless communication systems. If the battery drops below a minimum voltage, a radio or microprocessor will not reach the threshold voltage and brown out.

For example, the TI SensorTag C2650 has the following power characteristics:

- Standby mode: 0.24 mA
- Running with all sensors disabled: 0.33 mA
- LEDs
- All sensors on at 100 ms/sample data rate and broadcasting BLE: 5.5 mA:
 - Temperature sensor: 0.84 mA
 - Light sensor: 0.56 mA
 - Accelerometer and gyros: 4.68 mA
 - Barometric sensor: 0.5 mA

The TI SensorTag uses a standard CR2032 coin cell battery rated at 240 mAh. Therefore, the maximum life is expected to be about 44 hours. However, the rate of decline changes, and is not linear for battery-based devices, as we will see when we cover Peukert's capacity.

Many power management practices are employed, such as clock gating components not being used in silicon, reducing the clock rates of processors or microcontrollers, adjusting the sensing frequency and broadcast frequency, back-off strategies to reduce communication strength, and various levels of sleep modes. These techniques are widely used in the computing business as a general practice.

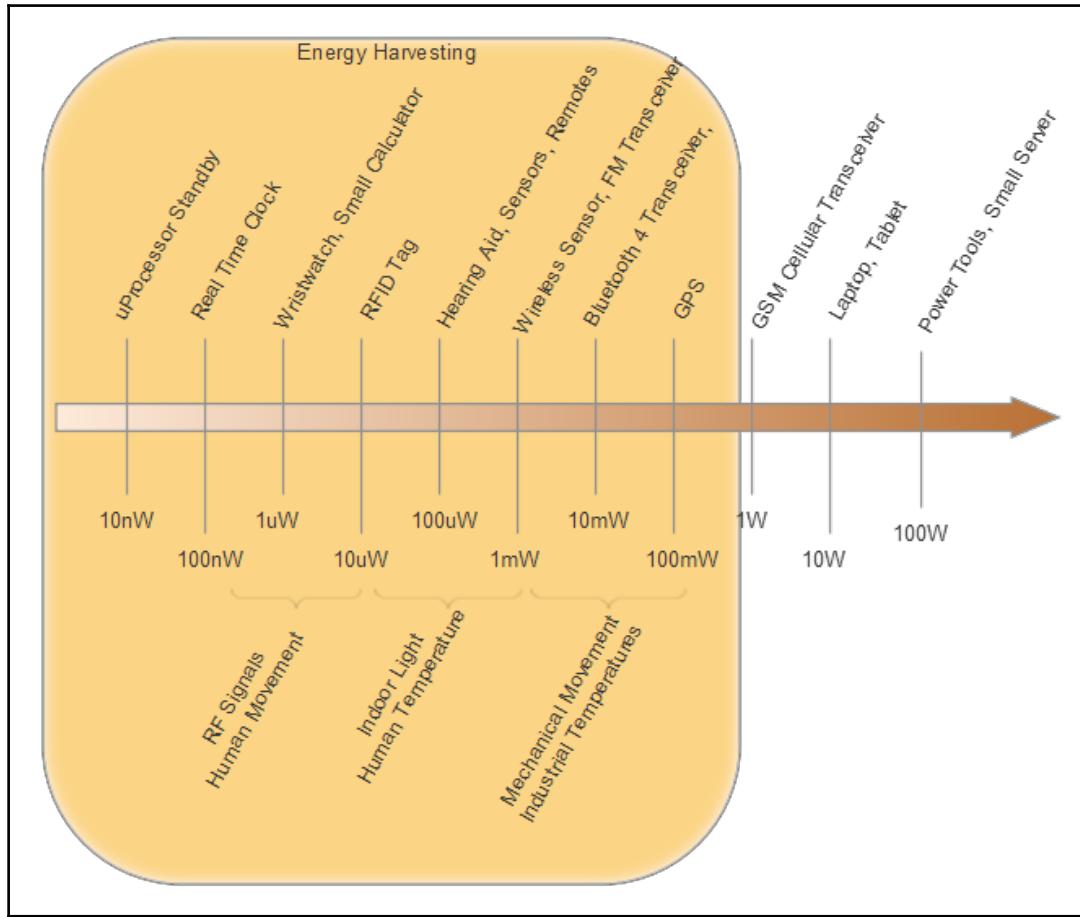


The techniques described here reflect reactionary power management techniques. They try to minimize energy usage based on dynamic voltage, frequency scaling, and other schemes. New techniques to consider on the horizon include approximate computing and probabilistic design. Both of these schemes rely on the fact that absolute precision is not necessary at all times in a sensor environment running at the edge, especially in use cases involving signal processing and wireless communication. Approximate computing can be done in hardware or software, and simply reduces the level of precision in integers when used with functional units such as addresses and multipliers (for example, the value 17,962 is fairly close to 17,970). Probabilistic design realizes many IoT deployments can tolerate certain degrees of faultiness to relax design constraints. Both techniques can reduce the number of gates and power to an almost exponential drop over regular hardware designs.

Energy harvesting

Energy harvesting is not a new concept, but is an important concept for IoT. Essentially, any system that represents a change in state (for example, hot to cold, radio signals, light) can convert its form of energy into electrical energy. Some devices use this as their sole form of energy, while others are hybrid systems that use harvesting to augment or extend the life of a battery. In turn, energy harvested can be stored and used (sparingly) to power low-energy devices such as sensors in IoT. Systems must be efficient in capturing energy and storing power. Therefore, advanced power management is needed. For example, if an energy harvesting system uses a piezoelectric mechanical harvesting technique embedded in a sidewalk, it will need to compensate when there is not enough foot traffic to keep the unit charged. Constant communication with energy harvesting systems can further drain the power. Typically, these IoT deployments will use advanced power management techniques to prevent complete loss of functionality. Techniques such as low standby currents, low-leakage circuits, and clock throttling are frequently used. The following figure illustrates the area where energy harvesting is ideal, and the technology that it can power. Care must be taken by the architect to ensure the system is not underpowered, nor overpowered.

Harvesting systems, in general, have a low energy potential and low conversion efficiency. An architect should consider energy harvesting in situations where there is a large supply of untapped waste energy, such as in industrial settings:

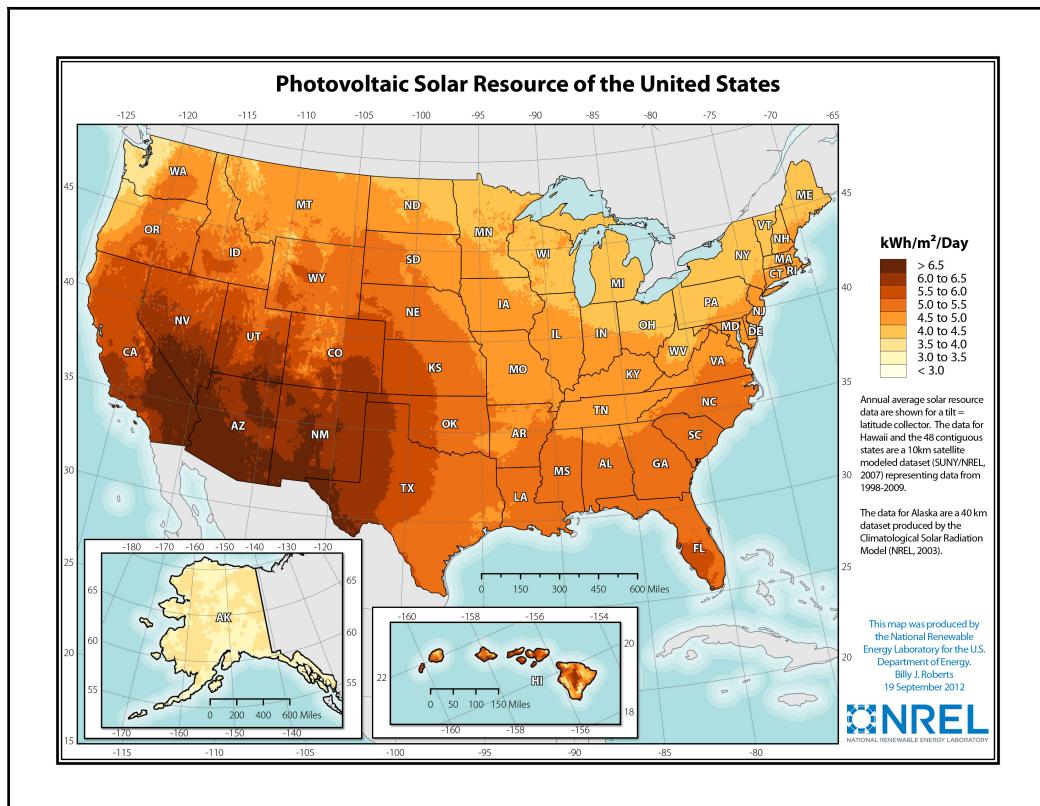


Energy harvesting sweet spots. The figure illustrates typical energy consumption for various devices.

Solar harvesting

Energy from light, whether it is natural or artificial, can be captured and used as an energy source. Earlier in this chapter, we discussed photodiodes and their relationship to sensing light. The same diode can be used in greater quantities to build a traditional solar array. The capacity of energy generation is a function of the area of the solar array. In practice, indoor solar generation is not as efficient as direct sunlight. Panels are rated by their maximum power output in the form of watts.

Solar harvesting is only as effective as how much the sun shines, which varies seasonally and geographically. A region such as the Southwestern US can reclaim considerable energy from direct photovoltaic sources. The Photovoltaic Solar Resource of the United States map was created by the National Renewable Energy Laboratory for the U.S. Department of Energy www.nrel.gov shown as follows:



Solar energy map of United States in energy density of kWh/m² 1998-2009.

In the United States, Southwest regions fare particularly well with sun intensity, generally lack cloud light barriers, and have good atmospheric conditions. Whereas, Alaska has the weakest energy density. Solar photovoltaics are not typically efficient. One can expect an 8% to 20% efficiency, with 12% being typical. Regardless, a 25 cm² solar array could produce 300 mW at peak power. Another factor is the incidence of light. For a solar collector to achieve such efficiency, the light source must be perpendicular to the array. If the angle of incidence changes as the sun moves, the efficiency drops further. A 12% efficiency collector when the sun is perpendicular will be roughly 9.6% efficient when the sun is 30 degrees from being perpendicular.

The most basic solar collector is the solar cell that is a simple *p-n* semiconductor, and similar to the photoelectric sensors discussed earlier. As explained earlier, an electric potential is generated between the *p* and *n* material when a photon is captured.

Piezo-mechanical harvesting

As mentioned earlier in this chapter, piezoelectric effects can be used as sensors, but they can also be used to generate power. Mechanical strains can be converted to energy through motion, vibration, and even sound. These harvesters could be used in smart roadways and infrastructures to harvest and change systems based on traffic movement, even when embedded in concrete. These devices produce currents on the order of milliwatts, and thus are suitable for very small systems with some form of energy collection and storage. This process can be performed using MEMS piezo-mechanical devices, electrostatic, and electromagnetic systems.

Electrostatic harvesting incorporates Faraday's law, which basically states that one can induce an electric current by changing the magnetic flux across a coil of wire. Here, the vibration is coupled either to the coil or a magnet. Unfortunately, this scheme in the IoT sensor area provides too little voltage for rectification.

Electrostatic systems use the change in distance between two capacitive plates held at a constant voltage or charge. As the vibration causes the distance to change between the plates, energy (*E*) can be harvested based on the following model:

$$E = \frac{1}{2} QV^2 = \frac{Q^2}{2C}$$

Here, Q is the constant charge on the plates, V is the constant voltage, and C represents capacitance in the preceding equation. Capacitance can also be represented by the length of the plate L_w , the relative static permittivity as ϵ_0 , and the distance between plates d as shown:

$$C = \epsilon_0 L_w / d$$

An electrostatic conversion has the advantage of being scalable and cost-efficient to produce through micromachining and semiconductor fabrication.

The last method for mechanical-to-electrical conversion is piezo-mechanical, which was talked about earlier in this chapter when discussing sensor input. The same basic concept applies to energy generation. As the piezo-mechanical MEMS device attempts to dampen the mass attached to it, the oscillations will be converted into an electrical current.

Another consideration for the capture and conversion of vibrational or mechanical energy is the need for conditioning before the energy is used or stored. Normally, a passive rectifier is used for conditioning by incorporating a large filtering capacitor. Other forms of energy harvesting do not need such conditioning.

RF energy harvesting

Radiofrequency (RF) energy harvesting has been in production for years in the form of RFID tags. RFID enjoys the benefit of being a near-field communication that uses a transceiver that essentially powers the RFID tag due to its close proximity.

For far-field applications, we need to harvest energy from broadcast transmissions. Broadcast transmissions are nearly everywhere, with service from televisions, cell signals, and radio. Capturing energy from radio frequencies versus other forms is particularly difficult, as RF signals have the smallest energy density of all harvesting techniques. The capturing of RF signals is based on having the appropriate antenna to capture a frequency band. Typical frequency bands used are in the 531 to 1611 kHz range (all in the AM radio range).

Thermal harvesting

Thermal energy can be converted into an electric current for any device exhibiting heat flow. Thermal energy can be converted into electrical energy by two basic processes:

- **Thermoelectric:** Direct conversion of thermal energy into electrical energy through the Seebeck effect.
- **Thermionic:** Also known as thermotunneling. Electrons are ejected from an electrode that is heated, and into an electrode that is cool.

The thermoelectric effect (Seebeck effect) is produced when a gradient of temperature exists in conducting materials. The flow of carriers from a hot to a cold region between two dissimilar electrical conductors creates a voltage differential. A thermocouple, or **thermoelectric generator (TEG)**, could effectively produce voltage simply based on the temperature difference of a human, based on their core body temperature and outside temperature. A temperature difference of 5 degrees Celsius could generate 40 uW at 3V. As heat flows through the conduction material, a hot-side electrode induces electron flow to a cold-side electrode-producing current. Modern thermoelectric devices use n or p-type bismuth telluride in series. One side is exposed to the source of heat (called the thermocouple), and the other is isolated. The energy harvested by the thermopile is proportional to the square of the voltage and equivalent to the temperature difference between the electrodes. One can model the energy harvested by a thermocouple by the following equation:

$$V = \int_{T_L}^{T_H} S_1(T) - S_2(T) dT$$

Here S_1 and S_2 represent the different Seebeck coefficients for each of the two materials (n and p-type) in the thermopile when there is a temperature differential, $T_H - T_L$. Since the Seebeck coefficients are functions of temperature and there exists a temperature difference, the result is a voltage difference. This voltage is generally very small, so many thermocouples are used in series to form a thermopile.

One substantial problem with current thermocouples is the poor efficiency of the energy conversion (less than 10%); however, their advantages are notable, including their small size and ease of manufacturing, resulting in fairly low costs. They also have a very long lifetime of over 100,000 hours. The main problem, of course, is finding a relatively constant source of thermal variance. Using such a device in an environment throughout multiple seasons and temperatures is challenging. For IoT devices, thermoelectric generation typically resides in the 50 mW range.

Thermionic generation is based on the ejection of electrons from a hot electrode to a cold electrode over a potential barrier. The barrier is the work function of the material, and is best used when there is a significant source of heat energy. While its efficiencies are better than thermoelectric systems, the energy required to jump the potential barrier makes it generally unsuitable for IoT sensor devices. Alternative schemes such as quantum tunneling could be considered, but it currently remains a research activity.

Energy storage

Typical storage for an IoT sensor will be a battery or supercapacitor. When considering the architecture for sensor power, one must consider several aspects:

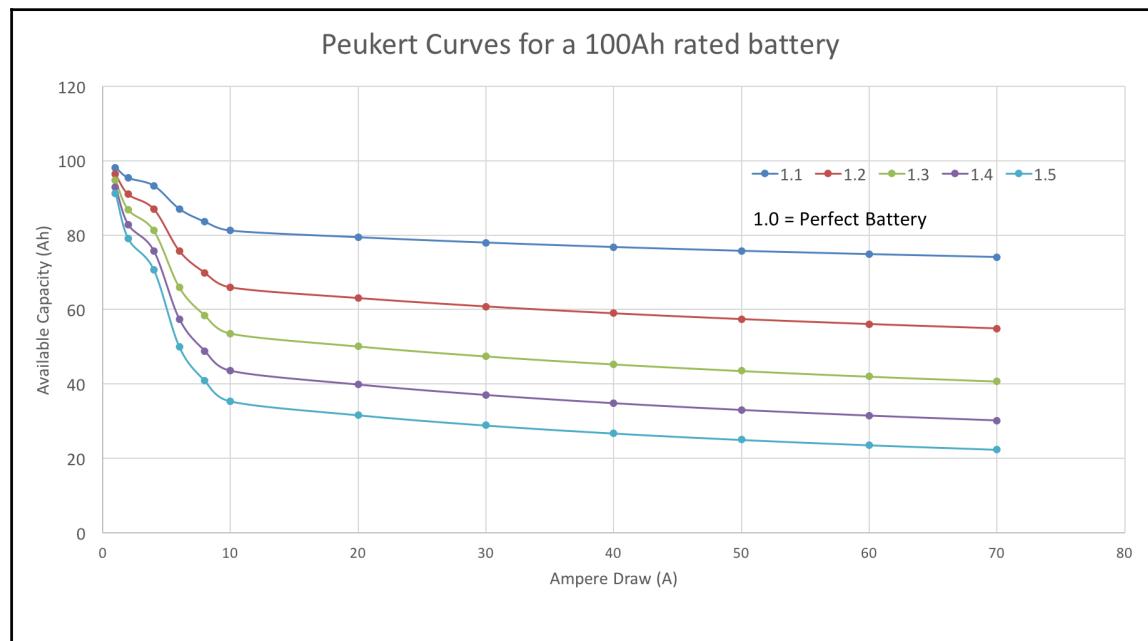
- Volume allowance for a power subsystem. Will a battery even fit?
- The battery energy capacity.
- Accessibility. If the unit is embedded in concrete, limited forms of energy regeneration can be used, and the difficulty of replacing batteries can be costly.
- Weight. Is the unit intended to fly as a drone, or float on water?
- How often will the battery be recharged?
- Is the renewable form of energy constantly available, or intermittent, as in solar?
- The battery power characteristics. How the battery's energy will vary over time as it's discharged.
- Is the sensor in a thermally constrained environment that can affect battery life and reliability?
- Does the battery have a profile that guarantees minimum current availability?

Energy and power models

Battery capacity is measured in amp-hours. A simplified equation to estimate the life of a battery power source is given as:

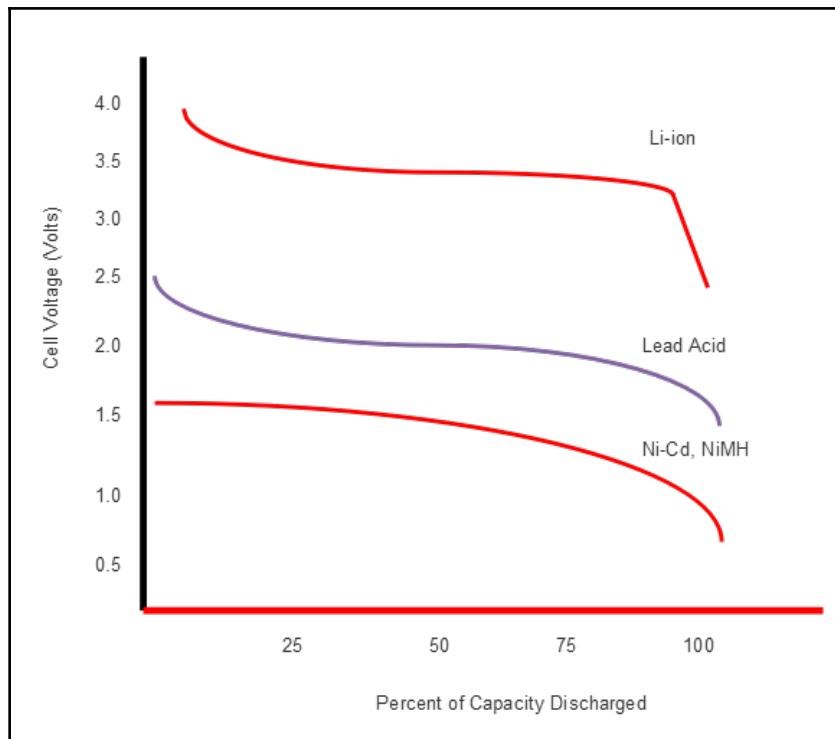
$$t = \frac{C_p}{I^n}$$

In the equation, C_p is Peukert's capacity, I represents the discharge current, and n is Peukert's exponent. The Peukert effect, as it's known, helps predict the life of a battery where the capacity of a battery decreases at a different rate as discharge increases. The equation shows how discharging at higher rates removes more power from the battery. Alternatively, discharging at lower rates will increase the effective runtime of the battery. One way of thinking of this phenomenon is a battery manufacturer rated at 100 Ah, and discharging it fully in 20 hours (5A, for this example). If one were to discharge it faster (say, in 10 hours), the capacity would be lower. If one were to discharge it more slowly (say, over 40 hours), it would be greater. However, when represented on a graph, the relationship is nonlinear. Peukert's exponent usually lies between 1.1 and 1.3. As n increases, we move further from a perfect battery to one that discharges faster as the current increases. Peukert's curve applies to lead-acid battery performance, and an example is shown in this next graph:



Peukert curves from 1.1 to 1.5 for a battery rated at 100A over 20 hours. The curves show the declining capacity as the Peukert coefficient grows.

One can see the difference in discharge rates for various types of batteries. One advantage of alkaline batteries is that the discharge rate is almost linear in a large portion of the graph. Lithium-ion has a stair-step function in performance, thus making battery charge predictions more difficult. That said, Li-ion provides a near steady and continuous voltage level over the life of the charge, and consistently powers electronics over the span of its charge:

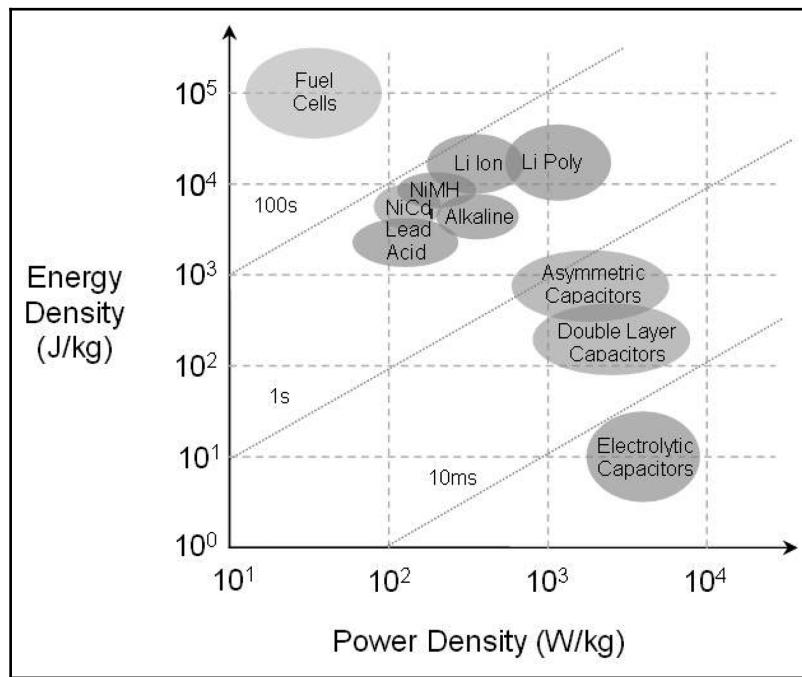


Example of relative discharge rates for various batteries. Li-ion provides a nearly constant voltage over its life, but has a precipitous drop towards the end of its storage capacity.

The graph also illustrates that lead acid and Ni-Cd have less voltage potential and a curvilinear degradation in power that can be computed more reliably. The trailing slopes are also indicative of Peukert's capacity.

Temperature greatly affects battery life, and specifically the electro-active carriers in a cell. As temperature increases, the internal resistance of a battery decreases when being discharged. Even when batteries are stored, they can self-discharge, which impacts the total lifetime of a battery.

A Ragone plot is a useful way to show the relationship between energy storage systems when trading off energy capacity and power handling. It is a log-based scale where the energy density (Wh/kg) of a power source is plotted against the power density (W/kg). This relationship shows devices that tend to have a greater lifetime (batteries) versus devices that tend to store more energy (supercapacitors):



Ragone plot depicting capacitor, supercapacitors, battery, and fuel cell differences in energy capacity versus lifetime supply of energy.

Batteries like Li-ion have higher energy density and discharge rates than nickel-cadmium and nickel-hydride batteries. Capacitors produce very high-power densities, but are relatively weak energy density. Note the plot is log-based, and also shows the discharge time for various storage systems. Image courtesy of C Knight, J. Davidson, S. Behrens "Energy Options for Wireless Sensor Nodes", Sensors, 2008, 8(12), 8037-8066.

Batteries

Typically, a **lithium-ion (Li-ion)** battery is the standard form of power in mobile devices due to its energy density. In such a battery, lithium-ions physically move from the negative electrode to the positive. During recharge, the ions move back to the negative region. This is known as an *ionic movement*.

Batteries also develop memories with many charge-discharge cycles. This capacity loss is expressed as a measure of the initial capacity (for example, 30% loss after 1,000 cycles). This degradation is almost directly correlated to environmental temperature, and loss will increase in a high-temperature environment. Therefore, it is imperative that the architects manage thermals in a constrained environment if lithium-ion is to be used.

Another factor in battery life is self-discharge. When unwanted chemical reactions occur in a battery cell, energy will be lost. The rate of loss depends on chemistry and temperature. Typically, a Li-ion can last for 10 years (at ~2% loss per month), while an alkaline battery will only last 5 years (15% to 20% loss per month).

Supercapacitors

Supercapacitors (or supercaps) store energy at significantly higher volumes than typical capacitors. A typical capacitor will have an energy density of somewhere between 0.01 Watt-hours/kg. A supercap has an energy density of 1 to 10 Watt-hours/kg, thus placing them closer to the energy density of a battery that can be on the order of 200 Watt-hours/kg. Like a capacitor, energy is stored electrostatically on a plate, and doesn't involve the chemical transfer of energy like a battery. Usually, supercaps are made of fairly exotic materials like graphene, which can impact on overall cost. Supercaps also have the advantage of charging to their full potential in seconds, whereas Li-ion batteries will charge within minutes to approximately 80%, but require a trickle current to safely get higher. Additionally, supercaps can't be overcharged, whereas Li-ion can be overcharged and can result in serious safety concerns. Supercaps come in two forms:

- **Electric double-layer capacitors (EDLC):** Use an activated carbon electrode and store energy electrostatically
- **Pseudocapacitors:** Use a transition metal oxide and electrochemical charge transfer

Supercaps have an advantage over batteries in predicting the remaining time power will be available. The remaining energy can be predicted from the terminal voltage, which changes over time. Lithium-ion batteries have a flat energy profile from fully charged to discharged, thus rendering time estimation difficult. Since the voltage profile of a supercap changes over time, a DC-DC converter is needed to compensate for wide variations in voltage.

In general, the main issues with supercaps or capacitors are the leakage current and the cost. As can be seen in the following table, supercaps have their place. One will often see them in a hybrid solution with regular batteries to provide instantaneous power (for example, electric vehicle acceleration), while the battery supply sustains running power.

Radioactive power sources

A radioactive source with a high energy density (10^5 kJ/cm^3) can generate thermal energy due to the kinetic energy of emitted particles. Sources such as cesium-137 have a half-life of 30 years and a power capacity of 0.015 W/gm. This method can generate power in the Watt-to-kW range, but isn't practical in low-power sensor levels for IoT deployments. Space vehicles have used this technology for decades. Promising developments using MEMS piezoelectronics that capture electrons and force a micro-armature to move can create mechanical energy that may be harvested. A secondary effect of radioactive decay is the relatively weak power density profile. A radiation source with a long half-life will have weaker power density. Thus, they are suitable for bulk-charging supercaps to provide momentary energy when needed. The final issue with radioactive sources is the significant weight of lead shielding required. Cesium-137 requires 80 mm/W of shielding, which can add significant cost and weight to an IoT sensor.

Energy storage summary and other forms of power

Choosing the correct power source is critical, as mentioned earlier. The following table provides a summary comparison of different components in a system to be considered when picking the correct power source:

Category	Li-ion Battery	Supercap
Energy density	200 Wh/kg	8-10 Wh/kg
Charge-discharge cycles	Capacity drops after 100 to 1,000 cycles	Nearly infinite
Charge-discharge time	1 to 10 hours	Milliseconds to seconds

Operational temperature	-20°C to +65°C	-40 degrees Celcius to +85 degrees Celsius
Operational voltage	1.2 V to 4.2 V	1 V to 3 V
Power delivery	Constant voltage over time	Linear or exponential decay
Charge rate	(Very slow) 40 C/x	(Very fast) 1,500 C/x
Operational life	0.5 to 5 years	5 to 20 years
Form factor	Very small	Large
Cost (\$/kWh)	Low (\$250 to \$1000)	High (\$10,000)

Summary

This chapter summarized several different sensors and endpoints used in IoT deployments. The IoT isn't about simply connecting a device to the internet. While that is a key component, the essence of IoT is connecting the analog world to the digital. Essentially, things and devices previously unconnected now have the opportunity to collect information and communicate it to other devices. This is powerful because data that had never been captured now has value. The ability to sense the environment will lead to more efficiencies, revenue streams, and value for customers. Sensing allows for smart cities, predictive maintenance, tracking assets, and analyzing the hidden meaning in massive aggregates of data. Powering such systems is also critical, and must be understood by architects. A poorly designed system can lead to too short a battery life, which will end in substantial costs to remediate.

The next chapter will bridge the endpoints to the internet through non-IP communication. We will explore various wireless personal area networks, and their various traits and functions that are prevalent in the IoT space.

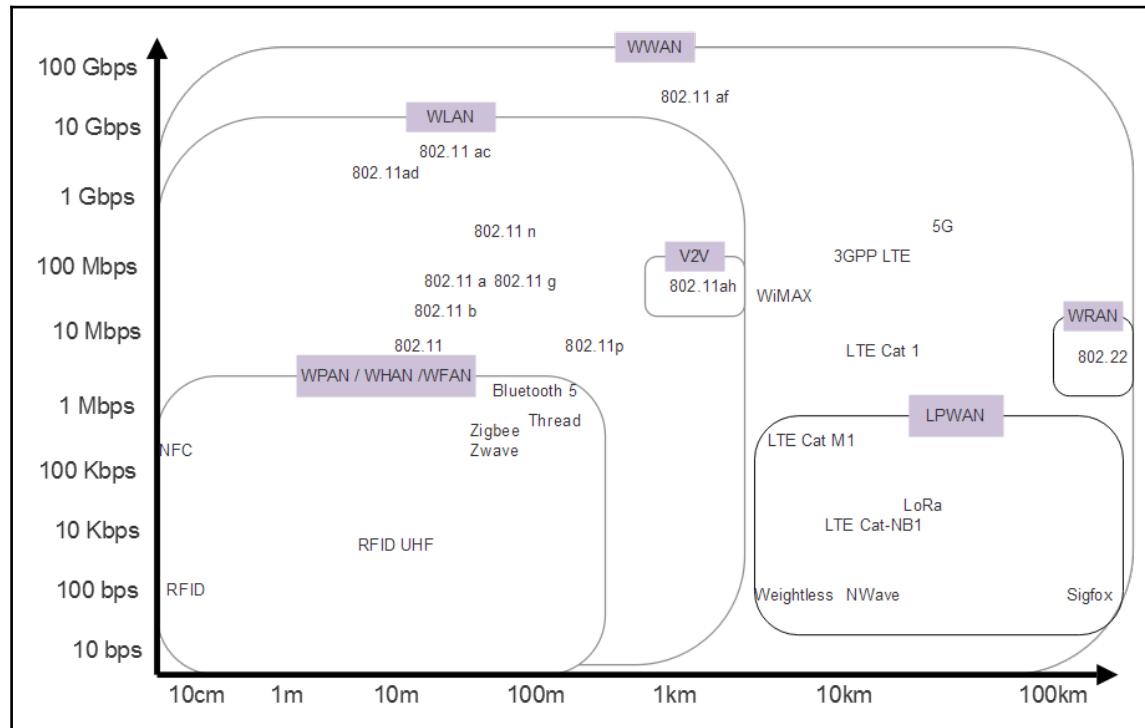
4

Communications and Information Theory

The IoT is more than simply the data from a sensor. We must first understand and architect the transport of moving sensor data from the remotest places on Earth to the cloud. There are a significant number of technologies and data paths to move data, and a bulk of the material in this book will investigate the aspects, constraints, and comparisons of communication choices for the architect.

We begin the WAN discussion with a review of wireless RF signals and the contributing factors that impact signal quality, limitations, interference, models, bandwidth, and range. There are many WAN communication protocols to choose from in different bands, and the architect must understand the pros and cons of choosing one radio spectrum over the other.

The following graphic helps delineate the various range and data rates for wireless protocols we will cover in later chapters. WPAN is often used with other near-range communication acronyms such as wireless **Field Area Network (FAN)**, **Wireless Local Area Network (WLAN)**, wireless **Home Area Network (HAN)**, wireless **Neighborhood Area Network (NAN)**, and **Wireless Body Area Network (WBAN)**:



Various wireless communication protocols and categories designed for different ranges, data rates, and use cases (power, vehicular, and so on.)

This chapter will provide foundational models and theory on communication systems, frequency spaces, and information theory. Communication constraints and comparison models will be provided to allow the architect to understand how and why certain types of data communication work, and where they won't work.

Next, we start with communication theory as it plays a fundamental part in choosing the correct mix of wireless technologies to deploy an IoT solution.

Communication theory

The IoT is a conglomeration of many disparate devices autonomously producing and/or consuming data on the very far edge of layers of networks and protocols. It is important to understand the constraints in building communication systems for the IoT or any form of networking. The internet of things will combine personal area networks, local area networks, and long-range wide area networks into a web of communication channels. Most of what makes the IoT possible is built around communication fabric; therefore, this chapter is dedicated to reviewing the fundamentals of network and communication systems. We will now focus on communication and signaling systems. We will explore the range, energy, and limitations of communication systems and how the architect will use these tools to develop a successful IoT solution.

RF energy and theoretical range

It is important when talking about wireless personal area networks, or any RF wireless protocol, to consider the range of transmission. Competing protocols use range, speed, and power as differentiators. As architects, we need to consider different protocols and design choices when implementing a complete solution. The range of transmission is based on the distance between transmitter and receiver antennas, the frequency of transmission, and the power of transmission.

The best form of RF transmission is unobstructed line-of-sight in a radio signal-free area. In most situations, this ideal model will not exist. In the real world, there are obstructions, reflections of signals, multiple wireless RF signals, and noise.

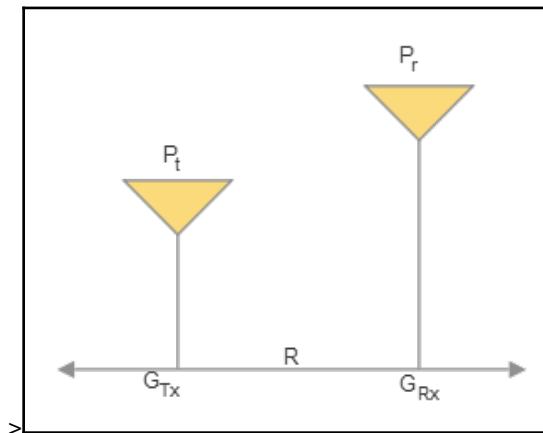
When considering a particular WAN and either a slower speed signal like 900 MHz versus a 2.4 GHz carrier signal, you can derive the attenuation of a function of wavelength for each frequency. This will provide guidance as to the signal strength at any range. The general form of the Friis transmission equation helps us here:

$$P_r = P_t G_{Tx} G_{Rx} \frac{\lambda^2}{(4\pi R)^2}$$

The **decibel (dB)** variant of the Friis equation is given as:

$$P_r = P_t + G_{Tx} + G_{Rx} + 20\log_{10}\left(\frac{\lambda}{4\pi R}\right)$$

Where G_{Tx} and G_{Rx} are the transmitter and receiver antenna gains, R is the distance between the transmitter and receiver, and P_r and P_t are the receiver and transmitter power respectively.



Friis equation graphical representation.

A 900 MHz signal at 10 meters will have a loss of 51.5 dB and a 2.4 GHz signal at 10 meters will have a 60.0 dB loss.

We can prove how power and range affect the quality of a signal by using a ratio referred to as the *link budget*. This is a comparison of the transmission power to sensitivity level and is measured in the logarithmic (dB) scale. One may simply want to increase power level to meet range requirements, but in many instances, this violates regulatory compliance or affects battery life. The other option is to improve the receiver's sensitivity level, which is exactly what Bluetooth 5 did in the latest specification. The link budget is given by the ratio of transmitter power over the receiver sensitivity as shown.

$$\text{LinkBudget} = \frac{\text{Tx Power}}{\text{Rx sensitivity level}}$$

The link budget is measured in the dB log scale; therefore, adding decibels is equivalent to multiplying the numeric ratios, which give us the equation:

$$\text{Receiver Power(dB)} = \text{Transmitted Power(dB)} + \text{Gains(dB)} - \text{Losses(dB)}$$

Assuming there is no factor contributing to any signal gains (for example, antenna gains), the only two ways of improving reception is increasing the transmission power or decreasing the loss.

When an architect must model the maximum range of a particular protocol, he or she will use the **Free Space Path Loss (FSPL)** formula. This is the amount of signal loss of an electromagnetic wave over a line of sight in free space (no obstacles). The contributing factors of FSPLF are the frequency (f) of the signal, the distance (R) between transmitter and receiver, and the speed of light (c). In terms of calculating FSPLF in decibels, the equation is:

$$\begin{aligned} FSPL(dB) &= 10\log_{10} \left(\left(\frac{4\pi R f}{c} \right)^2 \right) \\ &= 20\log_{10} \left(\frac{4\pi R f}{c} \right) \\ &= 20\log_{10} (R) + 20\log_{10} (f) + 20\log_{10} \left(\frac{4\pi}{c} \right) \\ &= 20\log_{10} (R) + 20\log_{10} (f) - 147.55 \end{aligned}$$

The FSPL formula is a simple first-order calculation. A better approximation takes into account reflections and wave interference from the earth ground plane, such as the *plane earth loss formula*. Here, h_t is the height of the transmitting antenna and h_r is the height of the receiving antenna. k represents the free space wave number, and is simplified as shown. We convert the equation to use dB notation:

$$\frac{Pr}{Pt} = L_{\text{plane earth loss}} \approx \left(\frac{\lambda}{4\pi R} k \frac{2h_t h_r}{R} \right) \approx \frac{h_t^2 h_r^2}{R^4} \text{ where } k = \frac{2\pi}{\lambda}$$

What is notable of the plane earth loss is the fact that the distance affects the loss by 40 dB per decade. Increasing the antenna height helps. The types of interference that can naturally occur include:

- **Reflections:** When a propagating electromagnetic wave strikes an object and results in multiple waves
- **Diffraction:** When a radio wave path between a transmitter and receiver is obstructed by objects with sharp edges
- **Scattering:** When the medium the wave travels through is comprised of objects smaller than the wavelength and the number of obstacles is large

This is an important concept since the architect must choose a WAN solution whose frequency balances the data bandwidth, the ultimate range of the signal, and the ability of the signal to penetrate objects. Increasing frequency naturally increases free space loss (for example, a 2.4 GHz signal has 8.5 dB less coverage than a 900 MHz signal). Generally speaking, 900 MHz signals will be reliable at twice the distance of 2.4 GHz signals. 900 MHz has a wavelength of 333 mm versus 125 mm for a 2.4 GHz signal. This allows a 900 MHz signal to have better penetration ability and not as much effect from scattering.

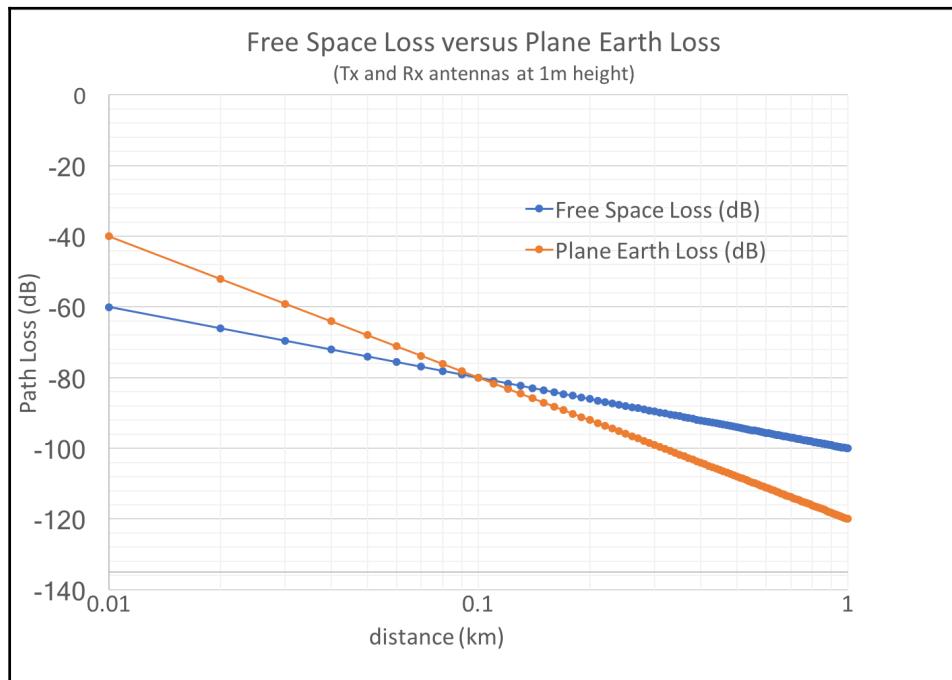
Scattering is a significant issue for WAN systems as many deployments do not have a free line of sight between antennas—instead the signal must penetrate walls and floors. Certain materials contribute differently to the attenuation of a signal.



A 6 dB loss equates to a 50% reduction in signal strength, while a 12 dB loss equates to a 75% reduction.

We see that 900 MHz has an advantage over 2.4 GHz in material penetration:

Material	Loss (dB) 900 MHz	Loss (dB) 2.4 GHz
0.25" Glass	-0.8 dB	-3 dB
Brick or Masonry Block Wall (8")	-13 dB	-15 dB
Sheet Rock	-2 dB	-3 dB
Solid Wood Door	-2 dB	-3 dB



Free Space Loss versus Plane Earth Loss (in dB) using a 2.4 GHz signal with antennas at 1 meter high.

As we will see, many protocols are commercially available and used globally in the 2.4 GHz spectrum. 2.4 GHz provides up to five times the data bandwidth as a 900 MHz signal and can have a much smaller antenna. Additionally, the 2.4 GHz spectrum is unlicensed and available for use in multiple countries.

	900 MHz	2.4 GHz
Signal strength	Generally reliable	Crowded band subject to interference
Distance	2.67x further than 2.4 GHz	Shorter, but can compensate with improved encoding (Bluetooth 5)
Penetration	Long wavelength can penetrate most materials and vegetation	Potential for interference with some building material
Data rate	Limited	About 2x to 3x faster than 900 MHz

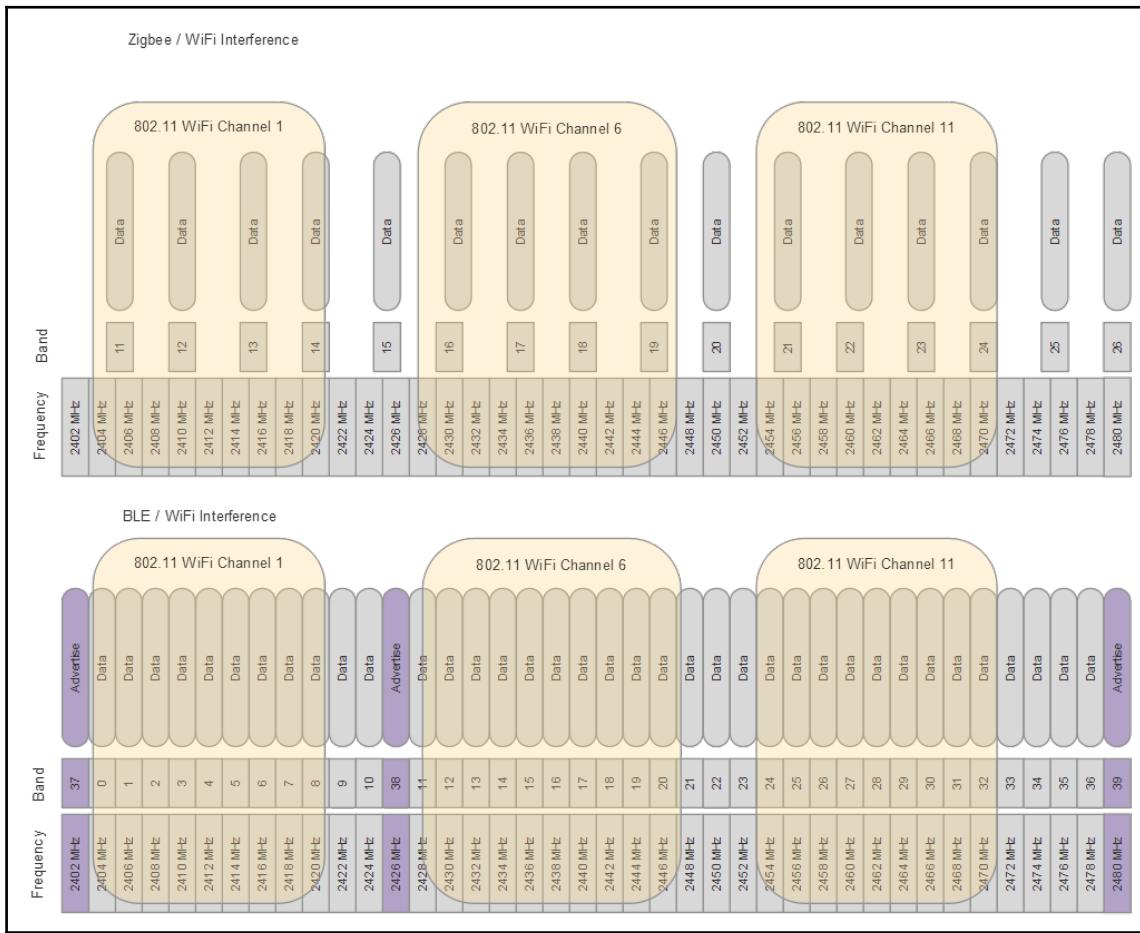
Signal interference	Signal can be affected by tall objects and obstructions, better through foliage	Less chance of channel interference with certain objects
Channel interference	Interference with 900 MHz cordless phones, RFID scanners, cell signals, baby monitors	Interference with 802.11 Wi-Fi
Cost	Medium	Low

These equations provide a theoretical model; no analytical equations give an accurate prediction for certain real-world scenarios such as multi-path losses.

RF interference

We will see throughout this chapter several novel schemes to reduce interference of signals. This is an issue for many forms of wireless technology as the spectrum is unlicensed and shared (we will discuss this more in the next section). Because of the fact that there may be several devices emanating RF energy in a shared space, interference will occur.

Take Bluetooth and 802.11 Wi-Fi; both operate in the shared 2.4 GHz spectrum but remain functional even in congested environments. **Bluetooth Low Energy (BLE)**, as we will see, will randomly select one of 40-2 MHz channels as a form of frequency hopping. We see in the following figure eleven free channels (three being advertising) on BLE that have a 15% chance of collision (especially since 802.11 does not hop between channels). The new Bluetooth 5 specification provides techniques such as slot availability masks to lock out Wi-Fi areas from the frequency hop list. Other techniques exist as well, which we will explore later. Here we show the ILM band for Zigbee and Bluetooth Low Energy. Also shown is a possible contention with three Wi-Fi channels in the 2.4 GHz spectrum.



Comparison of Bluetooth Low Energy (BLE) and Zigbee Interference with 802.11 Wi-Fi signals in the 2.4 GHz band. BLE provides more slots and frequency hopping to communicate in the event of Wi-Fi collisions.

Information theory

There are preliminary theories that need to be understood before detailing WAN specifics. Two areas that are germane to communication is how bitrate affects transmission power, which in turn affects range. There are limits to the integrity of data and bitrates, as we will learn. Additionally, we need to classify narrowband versus wideband communication.

Bitrate limits and the Shannon-Hartley theorem

In long-range communication and short-range communication, the goal is to maximize bitrate and distance within the constraints of spectrum and noise. The Shannon-Hartley theorem is composed of work from Claude Shannon of MIT in the 1940s (C. E. Shannon (1949/1998). *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press) and Ralph Hartley from Bell Labs in the 1920s (R. V. L. Hartley (July 1928).

"Transmission of Information" (PDF). *Bell System Technical Journal*). Foundational work was developed by Harry Nyquist, also of Bell Labs, who determined the maximum number of pulses (or bits) that could travel in a telegraph in a unit of time (H. Nyquist, *Certain Topics in Telegraph Transmission Theory*, in *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617-644, April 1928).

Essentially, Nyquist developed the sampling limit that determines how much theoretical bandwidth one has at a given sample rate. This is called the Nyquist rate, and is shown in the following equation:

$$f_p \leq 2B$$

Here, f_p is the pulse frequency and B is the bandwidth in Hertz. This states that the maximum bitrate is limited to twice the sampling rate. Looking at it another way, the equation identifies the minimum bitrate at which a finite bandwidth signal needs to be sampled to retain all information. Undersampling leads to aliasing effect and distortion.

Hartley then devised a way to quantify the information in what is called the line rate. The line rate can be thought of as bits per second (for example, Mbps). This is known as Hartley's law and is the precursor to Shannon's theorem. Hartley's law simply states the maximum number of distinguishable pulse amplitudes that can be transmitted reliably is limited by the dynamic range of the signal and the precision with which a receiver can accurately interpret each individual signal. Shown is Hartley's law in terms of M (number of unique pulse amplitude shapes) which is equivalent to the ratio of the number of voltage:

$$M = 1 + \frac{A}{\Delta V}$$

Converting the equation to a base-2 log gives us the line rate R :

$$R = f_p \log_2(M)$$

If we combine this with the preceding Nyquist's rate, we get the maximum number of pulses that can be transmitted over a single channel of bandwidth B . Hartley, however, did not work out with precision; the value of M (number of distinct pulses) could be affected by noise.

$$R \leq 2B \log_2(M)$$

Shannon reinforced Hartley's equation by considering the effects of Gaussian noise and completes Hartley's equation with a signal-to-noise ratio. Shannon also introduced the concept of error correction coding instead of using individually distinguishable pulse amplitudes. This equation is affectionately known as the Shannon-Hartley theorem:

$$C = B \log_2 \left(1 + \frac{S}{N} \right)$$

Here, C is the channel capacity in bits per second, B is the channel bandwidth in Hertz, S is the average received signal measured in watts, and N is the average noise on the channel measured in watts. The effect of this equation is subtle but important. For every decibel level increase in noise to a signal, capacity drops precipitously. Likewise, improving the signal-to-noise ratio will increase capacity. Without any noise, the capacity would be infinite.

It is also possible to improve the Shannon-Hartley theorem by adding a multiplier n to the equation. Here, n represents additional antennas or pipes. We have reviewed this previously as **multiple input, multiple output (MIMO)** technology.

$$C = B \times n \times \log_2 \left(1 + \frac{S}{N} \right)$$

To understand how Shannon's rule applies to the limits of wireless systems mentioned in this book, we need to express the equation in terms of energy per bit rather than the **signal-to-noise ratio (SNR)**. A useful example in practice is to determine the minimum SNR needed to achieve a certain bitrate. For example, if we want to transmit $C=200$ kbps over a channel with a bandwidth capacity of $B=5000$ kbps, then the minimum SNR required is given as:

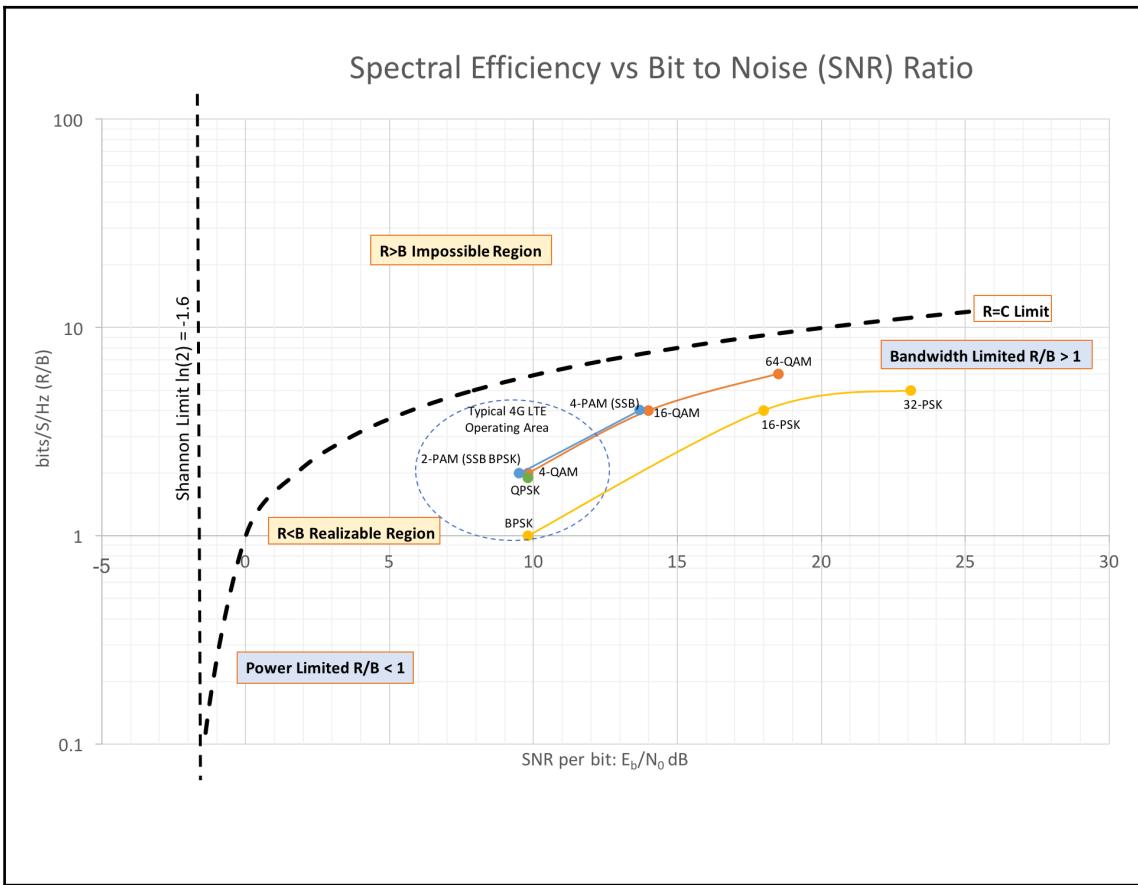
$$\begin{aligned} C &= B \log_2 \left(1 + \frac{S}{N} \right) \\ 200 &= 5000 \times \log_2 \left(1 + \frac{S}{N} \right) \\ \frac{S}{N} &= 0.028 \\ \frac{S}{N} &= -15.528dB \end{aligned}$$

This shows that it is possible to transmit data using a signal that is weaker than the background noise.

There is a limit to data rate, however. To show the effect, let E_b represent the energy of a single bit of data in joules. Let N_o represent the noise spectral density in watts/hertz. E_b/N_o is a dimensionless unit (however usually expressed in dB) that represents the SNR per bit, or commonly known as the **power efficiency**. Power efficiency expressions remove biases of modulation techniques, error coding, and signal bandwidth from the equation. We assume that the system is perfect and ideal such that the $R_b=C$ where R is the throughput. The Shannon-Hartley theorem can be rewritten as:

$$\begin{aligned}\frac{C}{B} &= \log_2\left(1 + \frac{E_b C}{N_0 B}\right) \\ \frac{E_b}{N_0} &= \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}} \\ \frac{E_b}{N_0} &\geq \lim_{\frac{C}{B} \rightarrow 0} \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}} = \ln(2) = -1.59dB\end{aligned}$$

This is known as the **Shannon limit** for an **Additive White Gaussian Noise (AWGN)**. An AWGN is a channel and is simply a basic form of noise used commonly in information theory to express the effects of random processes in nature. These noise sources are always present in nature and include things such as thermal vibrations, black body radiation, and the residual effects of the Big Bang. The "white" aspect of noise implies equal amounts of noise are added to each frequency. The limit can be drawn on a graph showing spectral efficiency versus SNR per bit:



Spectral efficiency to SNR (power efficiency) curve. The dotted line represents Shannon Limit which converges at $\ln(2)=-1.6$. Various modulation schemes are shown under the Shannon Limit with the typical range of 4G-LTE signals.

Regions of interest in the figure include the $R>B$ "Impossible Region". This region is above the Shannon limit of the curve. It states that no reliable form of information exchange can be above the limit line. The region below the Shannon limit is called the "Realizable Region" and is where $R<B$. Every protocol and modulation technique in any form of communication attempts to approach as close as possible to the Shannon limit. We can see where typical 4G-LTE using various modulation forms exist.

There are two other regions of interest. A "Bandwidth Limited" region toward the upper-right allows for high spectral efficiency and good E_b/N_o SNR values. The only constraint in this space is trading off a fixed or mandated spectral efficiency against the unconstrained power of transmission P , meaning the capacity has grown significantly over the available bandwidth. The opposite effect is called the **power limited** region towards the bottom-left of the chart. The Power Limited region is where the E_b/N_o SNR is very low, therefore Shannon's limit forces us down to low values of spectral efficiency. Here, we sacrifice spectral efficiency to get a given transmission quality P .



An example of a power limited use case is a space flight vehicle, such as the Cassini probe to Saturn. In that case, the free path space loss of signal is extremely large and the only way to get reliable data is to drop the data rate to very low values. We also see this in Bluetooth 5 using the new BLE Coded PHY. In that case, Bluetooth drops from 1 or 2 Mbps to as low as 125 kbps to improve range and data integrity.

The chart also shows some typical modulation schemes used today such as phase shifting, QAM, and others. The Shannon limit also shows that arbitrarily improving a modulation technique such as quadrature amplitude modulation for 4-QAM to 64-QAM doesn't scale linearly. The benefit of higher orders of modulation (for example, 4-QAM versus 64-QAM) is the fact that you can transmit more bits per symbol (two versus six). The main disadvantage with higher orders of modulation are:

- Using higher order modulations requires ever greater SNR to operate.
- Higher orders of modulation require much more sophisticated circuitry and DSP algorithms contributing to complexity
- Increasing the transfer of bits per symbol will increase the error rate



The Shannon theorem states that there is a maximum rate of information that can be transmitted over a communication channel in the presence of additive Gaussian white noise. As the noise decreases, the rate of information will increase but has an ultimate bound that cannot be breached. In any situation, if the transmission rate R is less than the channel capacity C, then there should be a method or technology to transmit the data without error.

Bit error rate

Another important characteristic of data transmission is the **Bit error rate (BER)**. BER refers to the number of bit errors received over a communication channel. BER is a unitless measurement expressed as a ratio or percentage. For example:

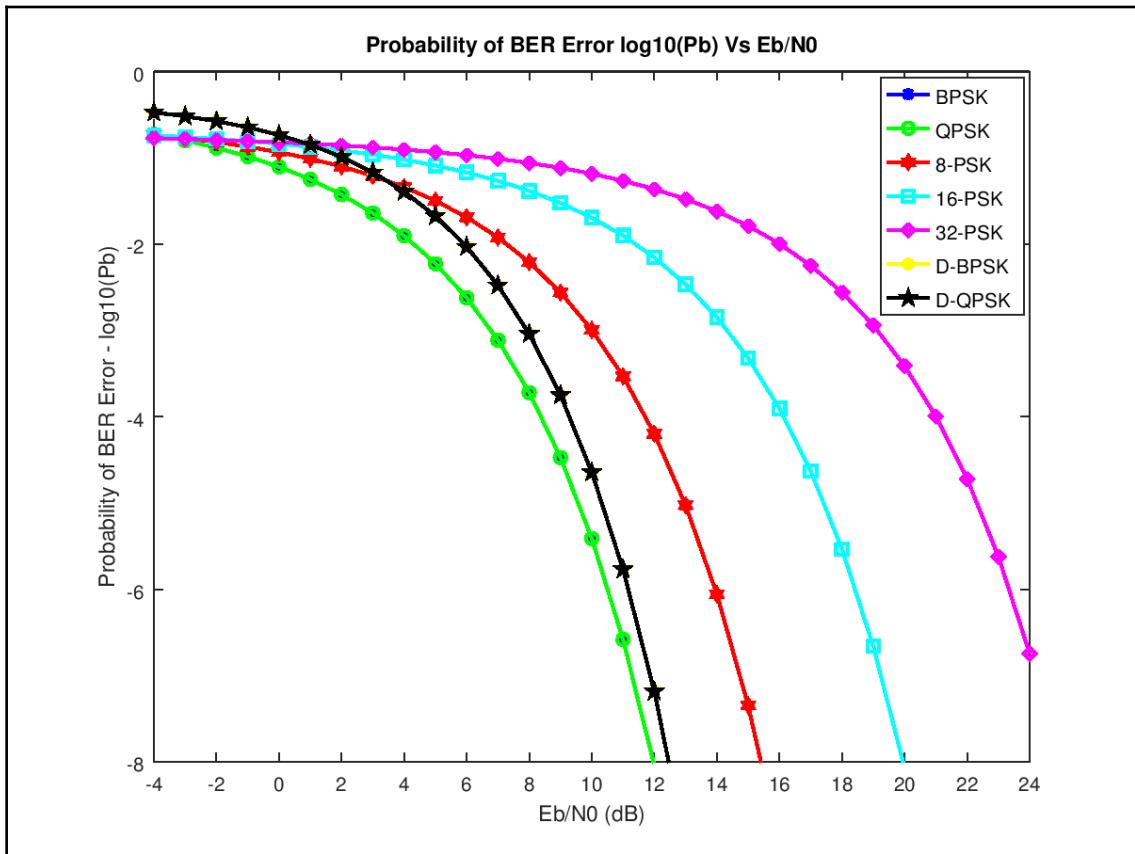
If an original transmission sequence is: 1 0 1 0 1 1 0 1 0 0

And the received sequence is: **0 0 1 0 1 **0 1 0 1 0**** (differences in bold)

Then the BER is 5 errors / 10 bits transferred = 50%

BER is affected by channel noise, interference, multipath fading, and attenuation. Techniques to improve BER include increasing transmission power, improving receiver sensitivity, using less dense/lower order modulation techniques, or adding more redundant data. The last technique is typically referred to as **forward error correction (FEC)**. FEC simply adds extra information to a transmission. In the most basic sense, one would add triple redundancy and a majority vote algorithm; however, this would reduce bandwidth by 3x. Modern FEC techniques include hamming codes and Reed-Solomon error correct codes. The BER can be expressed as a function of E_b/N_o SNR.

The following diagram shows a variety of modulation techniques and their respective BERs for various SNRs:



Bit error rates (Pb) versus power efficiency (E_b/N_0) SNR for various modulation schemes. As the SNR increases towards the right the BER naturally decreases.

What should be understood at this point is the following:

- We can now calculate the minimum SNR needed to achieve a certain data rate for a system
- The only way to add more capacity or bandwidth to a wireless service is to:
 - Add more spectrum and channel capacity, which improves the bandwidth linearly

- Add more antennas (MIMO), which improves the bandwidth linearly
- Improve the SNR with advanced antennas and receivers, which only improves the equation logarithmically
- The Shannon limit is the ultimate bound of digital transmission; exceeding the limit is possible, but the data integrity will be lost
- Factors that contribute to noise
- One cannot simply increase modulation levels without incurring costs to error rates and complexity

For 4G-LTE cellular signals, which are covered later in this book, it operates in the 700 MHz to 5 GHz spectrum with dozens of segregated bands within that range. A cell phone (or battery-based IoT device) has substantially less power than a cell tower, but it is often the case that an IoT device will be transmitting sensor data to the cloud. The uplink from the IoT device is what we examine here. The limit to uplink power is 200mW maximum, which is 23 dBm. This limits the overall range of the transmission; that limit, however, is dynamic and will vary based on the bandwidth of the channel and the data rate. 4G systems, like several WPAN and WLAN devices, use orthogonal frequency-division multiplexing. Each channel has many sub-carriers to address the multipath fading problems. If you sum all the data transmitting over sub-carriers, you achieve high data rates.

4G-LTE generally uses 20 MHz channels and LTE-A can use 100 MHz channels. These wide channels are limited by the overall spectrum of availability and are in contention with multiple carriers (ATT, Verizon, and so on) and other technologies sharing the spectrum. An additional complexity of cellular communication is that a carrier may have portions of the spectrum divided up and disjoint from each other.



Cat-3 LTE can use five, 10, or 20 MHz channels. The smallest channel granularity is 1.4 MHz. LTE-A is allowed to aggregate up to five 20-MHz channels for an aggregate bandwidth of 100 MHz.

A method to measure the distance a wireless device is functional is the **Maximum coupling loss (MCL)**. The MCL is the maximum distance at which total channel loss occurs between a transmitter and a receiving antenna but data service can still be delivered. MCL is a very common way to measure the coverage of a system. MCL will include antenna gains, path loss, shadowing, and other radio effects. Generally, a 4G-LTE system will have an MCL of about 142 dB. We will revisit MCL when examining the cellular IoT technologies such as Cat-M1.



What we should grasp at this point is that if we increase the listening time per bit, the noise level will go down. If we reduce the bitrate by 2x, then the following is true: $(\text{Bit_Rate} / 2) = (\text{Bit_Duration} * 2)$. Additionally, energy per bit increases by 2x and noise energy increases by $\sqrt{2}$. For example, if we reduce the Bit_Rate from 1 Mbps to 100 kbps, then Bit_Duration = increases by 10x. The range improves by the $\sqrt{10} = 3.162x$.

Narrowband versus wideband communication

Many of the wireless protocols that we will cover are known as wideband. We will see that the converse (narrowband) also has a place, especially for LPWAN. The differences between narrowband and wideband are as follows:

- **Narrowband:** A radio channel whose operational bandwidth does not exceed the channel's coherence bandwidth. Generally, when we talk about narrowband, we speak to signals that are 100 kHz or smaller in bandwidth. In narrowband, multipath causes amplitude and phase changes. A narrowband signal will fade uniformly, so adding more frequencies does not benefit the signal. Narrowband channels are also called flat fading channels because they usually will pass all spectral components with equal gain and phase to one another.
- **Wideband:** A radio channel whose operation bandwidth may significantly exceed its coherence bandwidth. These are generally greater than 1 MHz in bandwidth. Here, multipath causes the "self-interference" problem. Wideband channels are also called frequency-selective since different parts of the overall signal will be affected by the different frequencies in wideband. This is why wideband signals use a diverse range of frequencies to distribute power across many coherence bands to reduce fading effects.

Coherence time is a measure of the minimum time required for an amplitude or phase change to become uncorrelated from the previous value.

We have covered some forms of fading effects; however, there are many more. Path loss is a typical case where the loss is proportional to distance. Shadowing is where terrain, buildings, and hills create signal obstructions versus free space, and multipath fading occurs with the recombined scattering and wave interference of a radio signal on objects (due to diffraction and reflections). Other loss includes doppler shifts if the RF signal is in a moving vehicle. There are two categories of the fading phenomenon:

- **Fast fading:** This is characteristic of multipath fading when the coherence time is small. The channel will change every few symbols; therefore, coherence time will be low. This type of fading is also known as Rayleigh fading, which is the probability of random variances that cause an RF signal due to atmospheric particles or heavily built-up metropolitan areas.
- **Slow fading:** This occurs when the coherence time is long and there is movement over long distances usually due to doppler spreading or shadowing. Here, the coherence time is long enough to successfully transmit significantly more symbols than a fast fade path.

The following figure illustrates the difference between fast and slow fading paths.



Different RF Signal Fading Effects. From left to right: General path loss across a line of sight. Middle: Effects of slow fading due to large structures or terrains. Right: Combined effects of distance, slow fading, and fast fading.



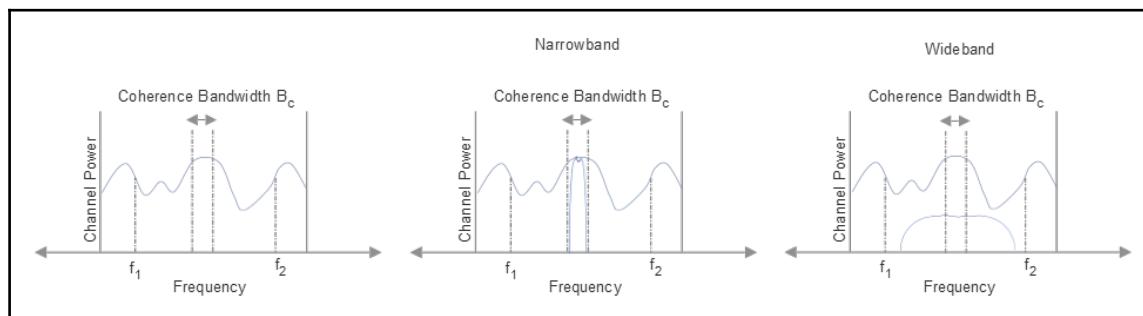
We will see that technologies employing narrowband signals use what is known as **time diversity** to overcome issues with fast fading. Time diversity simply means the signal and payload are transmitted multiple times in the hope that one of the messages gets through.

In a multipath scenario, the *delay spread* is the time between pulses from various multipath signals. Specifically, it is the delay between the first arrival of a signal and the earliest arrival of a multipath component of the signal.

The **coherence bandwidth** is defined as the statistical range of frequencies in which a channel is considered flat. This is a period of time when two frequencies are likely to have comparable fading. The coherence bandwidth B_c is inversely proportional to the delay spread D :

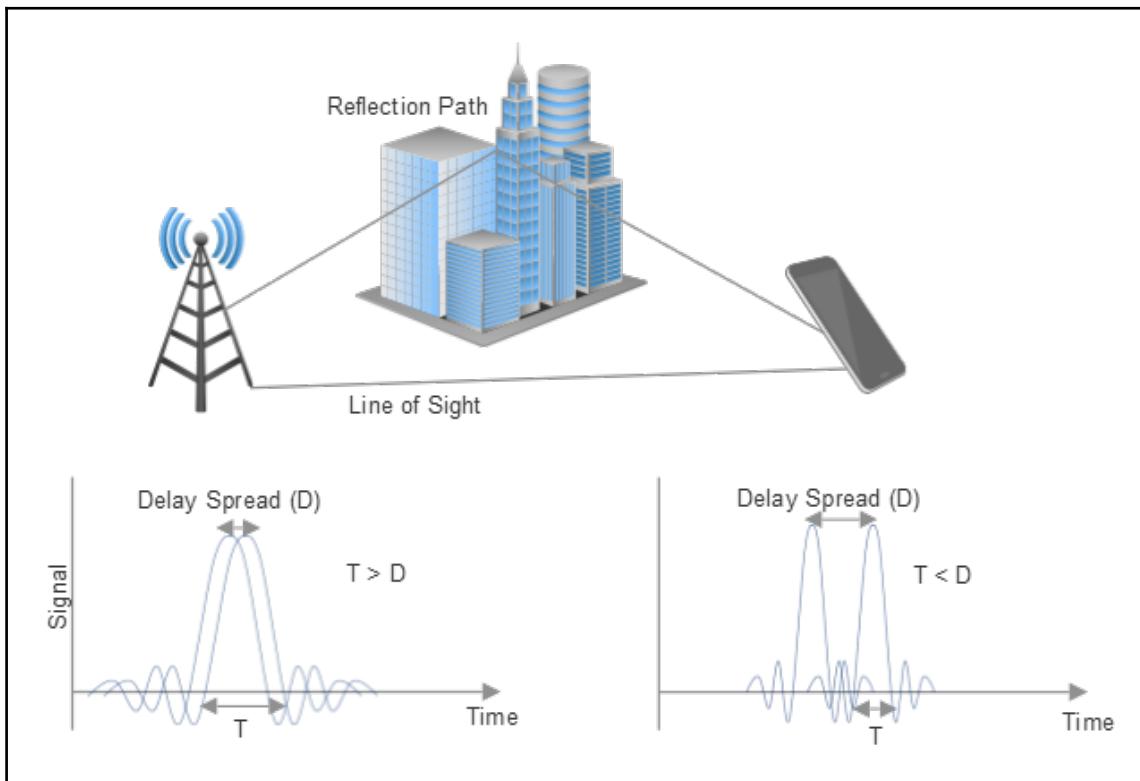
$$B_c \approx \frac{1}{D}$$

The time at which a symbol can be transmitted without intersymbol interference is $1/D$. The following figure illustrates coherence bandwidth for narrowband and wideband communication. Since wideband is larger than the coherence bandwidth B_c it is more likely to have independent fading attributes. That implies different frequency components will experience uncorrelated fading. Whereas, narrowband frequency components all fit within B_c and will experience uniform fading.



Coherence bandwidth and effects on narrowband and wideband. Frequencies f_1 and f_2 will fade independently if $|f_1 - f_2| > B_c$. Here narrowband is clearly shown to reside within B_c and wideband clearly exceeds the range of B_c by some margin.

One must ensure that the time between sending multiple signals from a multipath scenario is spread far enough as not to interfere with symbols. This is called **Inter-Symbol Interference (ISI)**. The following figure illustrates a delay spread being too short as to cause ISI. Given that the overall bandwidth $B \gg 1/T$ (where T is the pulse width time) and $B \gg B_c$ is implied, we can then generally state that bandwidth must be much larger than the coherence bandwidth: $B \gg B_c$.



Delay spread example. Here 2 signals from a form a multipath event. If the delay spread D is less than the pulse width T then the signals may not be spread far enough to override another multipath component. Whereas if the delay spread is large enough there may be no multipath conflict.



Generally, lower frequencies have a greater penetrating ability and less interference, but require larger antennas and have less available bandwidth for transmission. Higher frequencies have greater path loss but smaller antennas and more bandwidth.

The overall bitrate will be impacted by the delay spread. For example, say we use QPSK modulation and the BER is 10^{-4} , then for various delay spreads (D) we have:

- D= 256 μ S: 8 KBps
- D=2.5 μ S: 80 KBps
- D=100 ns: 2 Mbps

The radio spectrum

Wireless communication is based on radio waves and bands of frequencies within the overall radio spectrum. We will cover long-range communication in the next chapter for cellular and other long-range mediums, here we focus on the 1000 meter range or less. Here, we will look at the spectrum allocation process as well as the typical frequency uses for WAN devices.

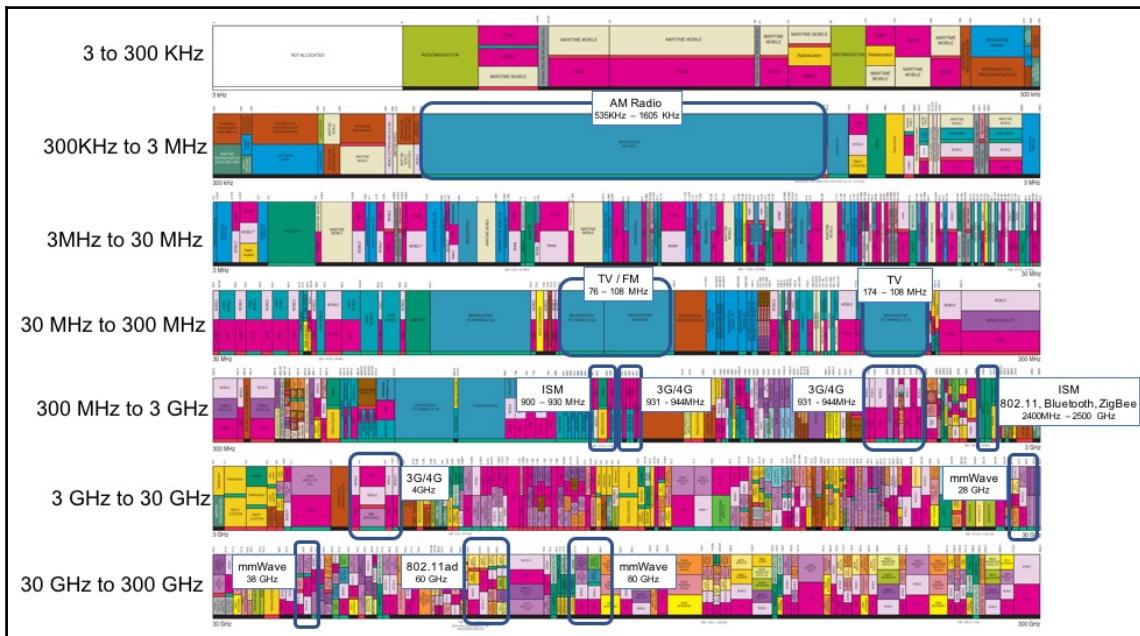
Governing structure

The spectrum ranges from 3 Hz to 3 THz, and allocation within the spectrum is governed by the **International Telecommunication Union (ITU)**. A band is considered a portion of the spectrum that can be allocated, licensed, sold, or freely used depending on the frequency. From an ITU perspective, the bands are categorized as follows:

Frequency	IEEE Band	European Union, NATO, US ECM	ITU	
			ITU Band	ITU Abbreviation
0.3 Hz				
3 Hz			1	ELF
30 Hz			2	SLF
300 Hz			3	ULF
3 kHz			4	VLF
30 kHz			5	LF
300 kHz			6	MF
3 MHz	HF		7	HF
30 MHz	VHF		8	VHF
250 MHz		B		
300 MHz	UHF	C		
500 MHz		D	9	UHF
1 GHz	L	E		
2 GHz	S	F		
3 GHz		G		
4 GHz	C	H		
6 GHz		I		
8 GHz	X	J		
10 GHz		K	10	SHF
12 GHz	Ku			
18 GHz	K			
20 GHz				
27 GHz	Ka			
30 GHz				
40 GHz	V	L		
60 GHz		M		
75 GHz	W			
100 GHz				
110 GHz	mm		11	EHF
300 GHz			12	THF
3 THz				

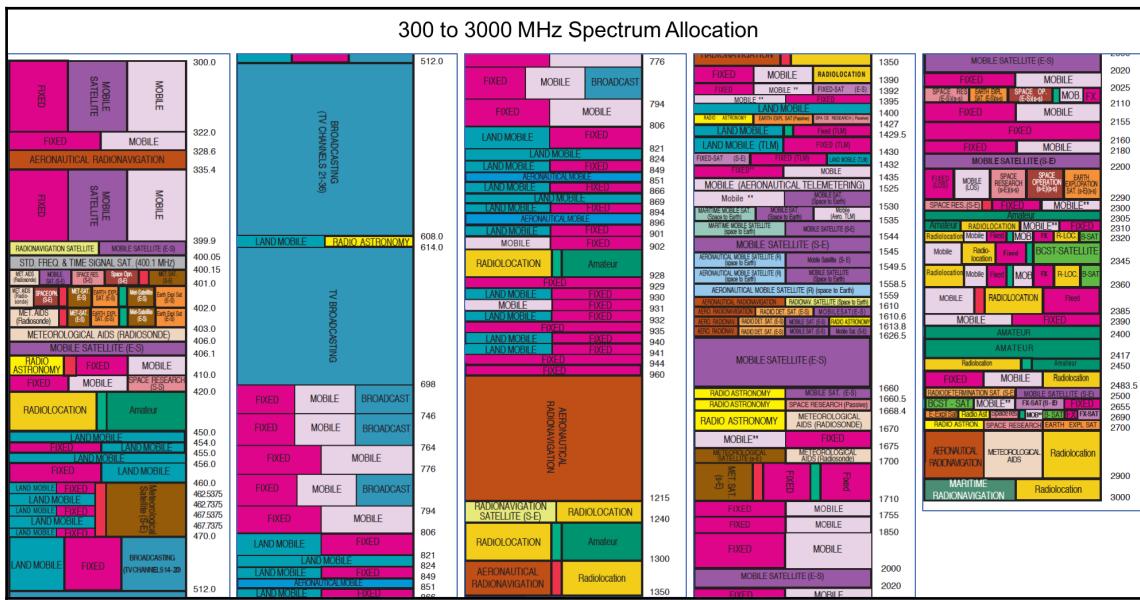
Within the United States, the **Federal Communications Commission (FCC)** and the **National Telecommunications and Information Administration (NTIA)** control the frequency spectrum usage rights. The FCC administers the non-Federal spectrum usage, while the NTIA administers the Federal usage (army, FAA, FBI, and so on).

The overall spectrum managed by the FCC ranges from the KHz spectrum through and including the GHz frequencies. The overall frequency distribution and allocation is shown below. Highlighted are the interesting frequencies that will be discussed in this book.



Full frequency allocation spectrum of the FCC with highlights of ranges covered in this book.

The following figure shows a small portion of the allocation of frequencies within the 900 MHz to 2.7 GHz range (common for WPAN signals) and how the frequencies are currently allocated and distributed. In many areas, the usage is multi-purpose and shared:



FCC and NTIA frequency allocation chart between 300 MHz and 3GHz. The chart represents a small portion of the overall frequency allocation. Source: FCC, "United States Frequency Allocations The Radio Spectrum", October 2003

The FCC also assigns frequencies in licensed and unlicensed spectrums. In the "unlicensed" or "licensed exempt" areas, users can operate without an FCC license but must use certified radio equipment and comply with technical requirements such as power limits and duty cycles. These are detailed in the FCC Part 15 Rules document. Users can operate within these spectrums but are subject to radio interference.

The licensed areas of the spectrum allow for exclusive usage for particular areas/locations. The allocation may be granted on a national basis or in discrete segments site by site. Since 1994, exclusivity and rights to these areas of the spectrum have been granted by auction for particular areas/segments/markets (for example, cellular market areas, economic areas, and so on). Some bands can be a hybrid of the two models, where bands may have been licensed on a site-by-site basis and later bands surrounding those licenses were auctioned for larger geographic or national areas. The FCC also allows for a secondary market and has established policies and procedures through spectrum leasing and transfer of control.

IoT deployments will typically use the licensed regions for long-range communication, which is covered in the next chapter. The unlicensed spectrum is typically used for **Industrial, Scientific and Medical (ISM)** devices. For IoT, IEEE 802.11 Wi-Fi, Bluetooth, and IEEE 802.15.4 protocols all reside in the 2.4 GHz unlicensed spectrum.

Summary

This chapter has provided foundation material to understand the theory and limitations of wireless communication. More information and deeper study are encouraged to understand the second and third order constraints of data transport. The architect should understand different models and constraints of wireless signals, RF energy dispersion, range, and the fundamental limits of information theory provided by Shannon's law. The architect should also understand the frequency space governance and allocation strategy. This chapter also provides a dialog and vernacular that will be reused in the upcoming chapters on WPAN, WLAN, and WAN.

The next chapter will start the journey of IoT data from the sensor to the cloud on its first hop across a near range personal area network. We build up from there to WLAN and WAN systems.

5

Non-IP Based WPAN

Sensors, and other things connected to the internet, need a method of transmitting and receiving information. This is the topic of **personal area network (PAN)** and near-range communication. In an IoT ecosystem, communication to a sensor or actuator can be a copper wire or a **Wireless Personal Area Networks (WPANs)**. In this chapter, we concentrate on the WPAN as that is the prevalent method for industrial, commercial, and consumer connections to the things of the internet. Wire-based connectivity is still used, but primarily in legacy industries and areas that are not radio-frequency friendly. There is a wide variety of different communication channels between the endpoint and the internet; some may be built on a traditional IP stack (6LoWPAN) and others use non-IP (internet protocol) communication to maximize energy savings (BLE).

We separate out IP and non-IP, as IP-based communication systems need further detail that non-IP communication doesn't necessarily need. Non-IP communication systems are optimized for cost and energy usage, whereas IP-based solutions usually have fewer constraints (for example, 802.111 Wi-Fi). The next chapter will detail the overlap of IP on the WPAN and WLAN.

This chapter will cover non-IP standards of communication, various topologies of WPAN networks (mesh, star), and constraints and goals of WPAN communication systems. These types of communication systems operate in the near meter to about 200-meter range (although some can reach much further). We will go deep into the **Bluetooth®** wireless protocol and the new Bluetooth 5.0 specification, as it sets a foundation to understand other protocols and is a prevalent and powerful part of IoT solutions.

This chapter will include technical details on proprietary and open standards. Each communication protocol has been adopted for certain reasons and use cases; they too will be covered in this chapter. Focus topics of this chapter include:

- RF signal quality and range
- Wireless spectrum allocation
- Bluetooth wireless protocol with special focus on the new Bluetooth 5 specification
- 802.15.4
- Zigbee®
- Z-Wave®

Wireless personal area network standards

This section will explore three relevant wireless personal area networks in the IoT space. A good portion of this section will be dedicated to Bluetooth as it provides a significant number of features and has a very deep presence in the IoT ecosystem. Additionally, Bluetooth 5.0 adds many features and abilities not seen before in the Bluetooth specification and provides range, power, speed, and connectivity that make it the strongest WPAN solution for many use cases. Zigbee, Z-Wave, and IEEE 802.15.4-based networks will also be investigated.

It is also helpful to know that the term WPAN is overloaded. Originally, it was to be a literal body and personal area network on a specific individual connected to wearable devices, but it has now extended its meaning.

802.15 standards

Many of the protocols and network models described in this section are based upon or have a foundation in the IEEE 802.15 working groups. The 802.15 group was initially formed to focus on wearable devices (coining the phrase personal area network). Their work has expanded significantly and now focuses on higher data rate protocols, meter to kilometer ranges, and specialty communications. Over one million devices are shipped each day using some form of 802.15.x protocol. The following is a list of the various protocols, standards, and specifications that the IEEE maintains and governs:

- **802.15**: Wireless personal area network definitions
- **802.15.1**: Original foundation of the Bluetooth PAN

- **802.15.2:** Coexistence specifications for WPAN and WLAN for Bluetooth
- **802.15.3:** High data rate (55 Mbps+) on WPAN for multimedia
 - **802.15.3a:** High-speed PHY enhancements
 - **802.15.3b:** High-speed MAC enhancements
 - **802.15.3c:** High-Speed (>1 Gbps) using mm-wave (millimeter wave) technology
- **802.15.4:** Low data rate, simple, simple design, multi-year battery life specifications (Zigbee)
 - **802.15.4-2011:** Rollup (specifications a-c) includes UWB, China, and Japan PHYs
 - **802.15.4-2015:** Rollup (specifications (d-p) includes RFID support, medical-band PHY, low energy, TV white spaces, rail communications
 - **802.15.4r (on hold):** Ranging protocol
 - **802.15.4s: Spectrum Resource Utilization (SRU)**
 - **802.15.t:** High rate PHY of 2 Mbps
- **802.15.5:** Mesh networking
- **802.15.6:** Body area networking for medical and entertainment
- **802.15.7:** Visible light communications using structured lighting
 - **802.15.7a:** Extends range to UV and near-IR, changed name to optical wireless
- **802.15.8: Peer Aware Communications (PAC)** infrastructure-less peer to peer at 10 Kbps to 55 Mbps
- **802.15.9: Key Management Protocol (KMP),** management standard for key security
- **802.15.10:** Layer 2 mesh routing, recommend mesh routing for 802.15.4, multi PAN
- **802.15.12:** Upper layer interface, attempts to make 802.15.4 easier to use 802.11 or 802.3

The consortium also has working groups investigating dependability (IG DEP) to address wireless reliability and resilience, high data rate communications (HRRC IG), and terahertz communications (THz IG).

Bluetooth

Bluetooth is a low-power wireless connectivity technology used pervasively in technology from cell phones sensors, and keyboards to video game systems. The name Bluetooth refers to King Harald Blatand in the region of what is now Norway and Sweden in around 958AD. King Blatand got his name from his liking of blueberries and/or the eating of his frozen enemies. Regardless, Bluetooth is derived from his name because King Blatand brought together warring tribes, and the same could be said for the formation of the initial Bluetooth SIG. Even the Bluetooth logo is a combination of runes from an ancient Germanic alphabet used by the Danes. Today, Bluetooth is prevalent and this section will focus on the new Bluetooth 5 protocol ratified by the Bluetooth SIG in 2016. Other variants will be called out as well.

This section will detail Bluetooth technology with special emphasis on the new Bluetooth 5.0 specification. To learn more about older Bluetooth technologies, refer to the Bluetooth SIG at www.bluetooth.org.

Bluetooth history

Bluetooth technology was first conceived at Ericsson in 1994 with the intent to replace the litany of cables and cords connecting computer peripherals with an RF medium. Intel and Nokia also joined in with the intent to wirelessly link cell phones to computers in a similar manner. The three formed an SIG in 1996 at a conference held at the Ericsson plant in Lund, Sweden. By 1998, there were five members of the Bluetooth SIG: Intel, Nokia, Toshiba, IBM, and Ericsson. That year, version 1.0 of the Bluetooth specification was released. Version 2.0 was later ratified in 2005 when the SIG had over 4000 members. In 2007, the Bluetooth SIG worked with Nordic Semiconductor and Nokia to develop Ultra Low Power Bluetooth, which now goes by the name **Bluetooth Low Energy (BLE)**. BLE brought an entirely new segment to the market in devices that could communicate using a coin cell battery. By 2010, the SIG released the Bluetooth 4.0 specification, which formally included BLE. Currently, there are over 2.5 billion shipping Bluetooth products and 30,000 members in the Bluetooth SIG.

Bluetooth has been used extensively in IoT deployments for some time, being the principal device when used in low energy mode (LE) for beacons, wireless sensors, asset tracking systems, remote controls, health monitors, and alarm systems.

Throughout its history, Bluetooth and all the optional components have been under GPL license and are essentially open source.

The revision history of Bluetooth as it has grown in features and abilities is shown in the following table:

Revision	Features	Release date
Bluetooth 1.0 and 1.0B	Basic rate Bluetooth (1 Mbps) Initial version released.	1998
Bluetooth 1.1	IEEE 802.15.1-2002 standardized 1.0B specification defects resolved Non-encrypted channel support Received Signal Strength Indicator (RSSI)	2002
Bluetooth 1.2	IEEE 802.15.1-2005 Rapid connection and discovery Frequency hopping spread spectrum (AFH) Host controller interface (three-wire UART) Flow control and retransmission modes	2003
Bluetooth 2.0 (+EDR optional)	Enhanced Data Rate Mode (EDR): 3 Mbps	2004
Bluetooth 2.1 (+EDR optional)	Secure Simple Pairing (SSP) using public key cryptography with four unique authentication methods Extended Inquiry Response (EIR) allows for better filtering and reduced power	2007
Bluetooth 3.0 (+ EDR optional) (+HS optional)	L2CAP enhanced retransmission mode (ERTM) for reliable and unreliable connection states Alternate MAC/PHY (AMP) 24 Mbps using 802.11 PHY Unicast connectionless data for low latency Enhanced power control	2009
Bluetooth 4.0 (+ EDR optional) (+HS optional) (+LE optional)	AKA BluetoothSmart Introduced Low Energy mode (LE) Introduced ATT and GATT protocols and profiles Dual mode: BR/EDR and LE mode Security manager with AES encryption	2010
Bluetooth 4.1	Mobile wireless service (MWS) coexistence Train nudging (coexistence feature) Interlaced scanning (coexistence feature) Devices support multiple simultaneous roles	2013
Bluetooth 4.2	LE secure connections Link layer privacy IPv6 support profile	2014
Bluetooth 5.0	Slot availability masks (SAM) 2 Mbps PHY and LE LE long range mode LE extended advertising modes Mesh networking	2016

Bluetooth 5 communication process and topologies

Bluetooth wireless is comprised of two wireless technology systems: **Basic Rate (BR)** and **Low Energy (LE or BLE)**. Nodes can either be advertisers or scanners by this definition:

- **Advertiser:** Devices transmitting advertiser packets
- **Scanner:** Devices receiving advertiser packets without the intention to connect
- **Initiator:** Devices attempting to form a connection

There are several Bluetooth events that transpire in a Bluetooth WPAN:

- **Advertising:** Initiated by a device to broadcast to scanning devices to alert them of the presence of a device wishing to either pair or simply relay a message in the advertisement packet.
- **Connecting:** This event is the process of pairing a device and a host.
- **Periodic advertising:** (for Bluetooth 5) allows an advertising device to periodically advertise over the 37 non-primary channels by channel hopping at an interval of 7.5ms to 81.91875s.
- **Extended advertising:** (for Bluetooth 5) allows for extended PDUs to support advertisement chaining and large PDU payloads, possibly as well as new use cases involving audio or other multimedia (covered in the *Beaconing* section of this chapter).

In LE mode, a device may complete an entire communication by simply using the advertising channel. Alternatively, communication may require pair-wise bi-directional communication and force the devices to formally connect. Devices that must form this type of connection will start the process by listening to advertising packets. The listener is called an initiator in this case. If the advertiser issues a connectable advertising event, the initiator can make a connection request using the same PHY channel it received the connectable advertising packet on.

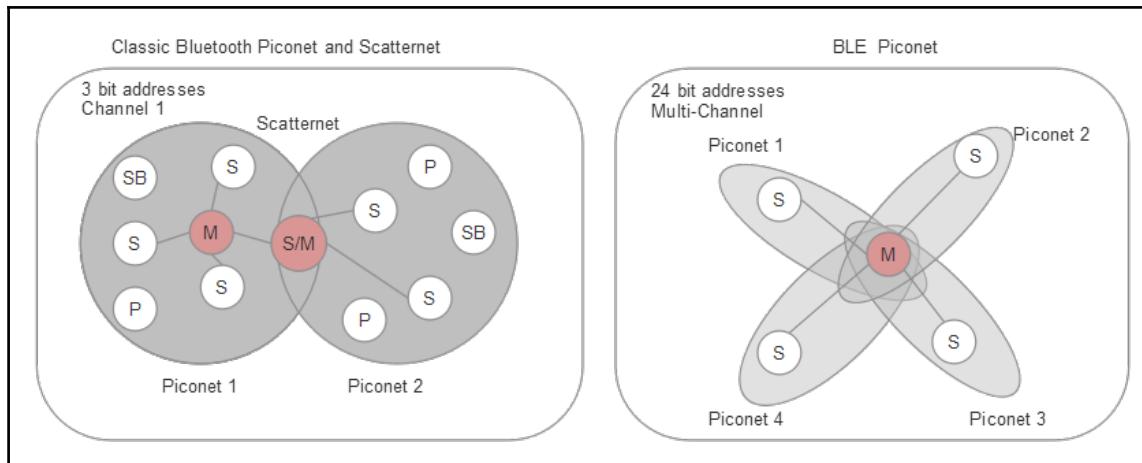
The advertiser can then determine if it wishes to form a connection. If a connection is formed, the advertising event ends and the initiator is now called the **master** and the advertiser is called the **slave**. This connection is termed a **piconet** in Bluetooth jargon and connection events transpire. The connection events all take place on the same starting channel between the master and slave. After data has been exchanged and the connection event ends, a new channel can be chosen for the pair using frequency hopping.

Piconets form in two different fashions depending on BR/EDR mode or BLE mode. In BR/EDR, the piconet uses three-bit addressing and can only reference seven slaves on one piconet. Multiple piconets can form a union and then be called a **scatternet**, but there must be a second master to connect to and manage the secondary network. The slave/master node takes on the responsibility of bridging two piconets together. In BR/EDR mode, the network uses the same frequency hopping schedule and all the nodes will be guaranteed to be on the same channel at a given time. In BLE mode, that system uses 24-bit addressing so the number of slaves associated with a master is in the millions. Each master-slave relationship is itself a piconet and can be on a unique channel. In a piconet, nodes may be a **master (M)**, **slaves (S)**, **standby (SB)**, or **parked (P)**. Standby mode is the default state for a device. In this state, it has the option to be in a low-power mode. Up to 255 other devices can be in an SB or P mode on a single piconet.



Note that Bluetooth 5.0 has deprecated and removed parked states in piconets; only Bluetooth devices up to version 4.2 will support a parked state. Standby state is still supported by Bluetooth 5.0.

A piconet topology is illustrated in the following diagram:



The difference between classic (BR/EDR) Bluetooth and BLE piconets. In BR/EDR mode up to seven slaves can be associated on a single piconet due to 3-bit addressing. They all share a common channel between the seven slaves. Other piconets can join the network and form a scatternet only if an associated master on the secondary network is present. In BLE mode millions of slaves can join in multiple piconets with a single master due to 24-bit addressing. Each piconet can be on a different channel but only one slave can associate with the master in each piconet. Practically speaking, BLE piconets tend to be much smaller.

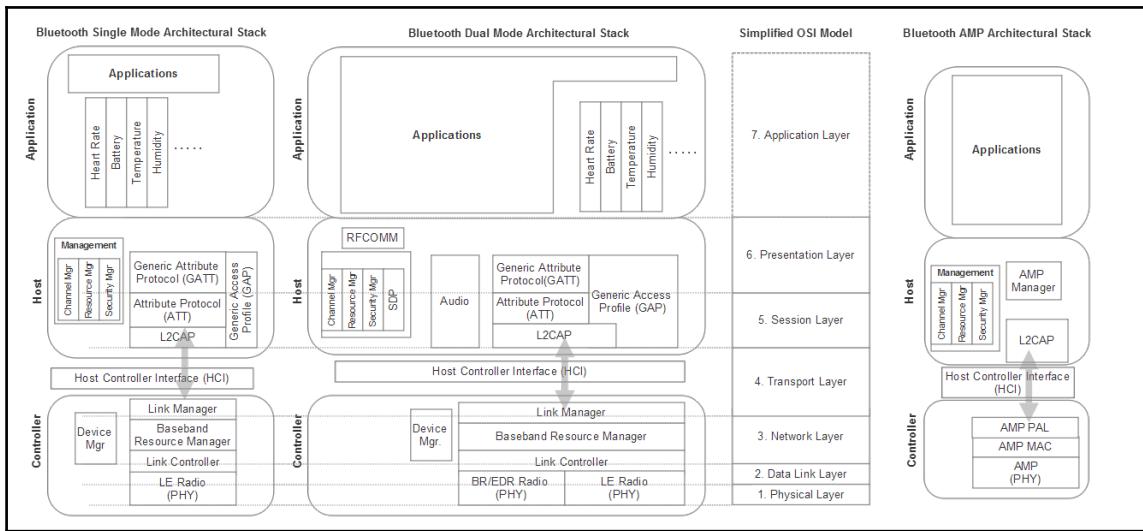
Bluetooth 5 stack

Bluetooth has three basic components: a hardware controller, host software, and application profiles. Bluetooth devices come in single and dual-mode versions, which means they either support only the BLE stack or they support classic mode and BLE simultaneously. In the following figure, one can see the separation between the controller and host at the **Host Controller Interface (HCI)** level. Bluetooth allows for one or more controllers to be associated with a single host.

The stack consists of layers, or protocols and profiles:

- **Protocols:** Horizontal tiers and layers representing functional blocks. The following diagram represents a stack of protocols.
- **Profiles:** Represent vertical functions that use protocols. Profiles will be detailed in a later section, but cover generic attribute profiles and generic access profiles.

The following figure represents a comprehensive architectural diagram of the Bluetooth stack. Including BR/EDR and BLE modes as well as AMP mode.



Bluetooth Single Mode (BLE only) and Dual Mode (Classic and BLE) versus a comparison to simplified OSI stack. The right-side diagram illustrates the AMP mode. Note the separation of responsibilities between the host software platform with the upper stack and the controller hardware for the bottom stack. HCI is the transport channel between the hardware and the host.

There are essentially three Bluetooth modes of operation shown in the preceding figure (each requiring a different PHY):

- **Low Energy (LE) mode:** This uses the 2.4 GHz ISM band and employs FHSS for interference protection. The PHY differs from BR/EDR and AMP radios by modulation, coding, and data rates. LE operates at 1 Msym/s at a bit rate of 1 Mbps. Bluetooth 5 allows for multiple configurable data rates of 125 Kbps, 500 Kbps, 1 Mbps, and 2 Mbps (more on this later).
- **Basic Rate/Enhanced Data Rate mode (BR/EDR):** This uses a different radio than LE and AMP but operates in the ISM 2.4 GHz band. Basic radio operation is rated at 1 Msym/s and supports a bit rate of 1 Mbps. EDR sustains a data rate of 2 or 3 Mbps. This radio uses FHSS for interference protection.
- **Alternative MAC/PHY (AMP):** This is an optional feature that uses 802.11 for high-speed transport up to 24 Mbps. This mode requires both the master and slave device to support AMP. This is a secondary physical controller, yet requires the system to have a BR/EDR controller to establish initial connection and negotiation.

We will now detail the function of each element of the stack. We start with the common blocks of BR/EDR and LE then list details for AMP. In all three cases, we will start with the physical layer and move up the stack towards the application layer.

Core architectural blocks:

- Controller level:
 - **BR/EDR PHY (controller block)**: Responsible for transmitting and receiving packets through a physical channel on 79 channels.
 - **LE PHY**: Low energy physical interface responsible for managing 40 channels and frequency hopping.
 - **Link controller**: Encodes and decodes Bluetooth packets from the data payload.
 - **Baseband resource manager**: Responsible for all access to the radio from any source. Manages the scheduling of physical channels and negotiates access contracts with all entities to ensure QoS parameters are met.
 - **Link manager**: Creates, modifies, and releases logical links and updates parameters related to physical links between devices. It is reused for BR/EDR and LE modes using different protocols.
 - **Device manager**: Block in the controller baseband level that controls the general behavior of Bluetooth. Responsible for all operations not related to data transmission, including making devices discoverable or connectable, connecting to devices, and scanning for devices.
 - **Host Controller Interface (HCI)**: This is a separation between the host and the silicon controller in layer four of the network stack. It exposes interfaces to allow a host to add, remove, manage, and discover devices on the piconet.
- Host level:
 - **L2CAP**: This is the logical link control and adaptation protocol. It is used to multiplex logical connections between two different devices using higher level protocols than the physical layer. It can segment and reassemble packets.
 - **Channel manager**: Responsible for creating, managing, and closing L2CAP channels. A master will use the L2CAP protocol to communicate to a slave channel manager.

- **Resource manager:** Responsible for managing the ordering of submission of fragments to the baseband level. Helps ensure the quality of service conformance.
- **Security Manager Protocol (SMP):** Also known as security manager protocol. This block is responsible for generating keys, qualifying keys, and storing keys.
- **Service Discovery Protocol (SDP):** Discovers services offered on other devices by UUID.
- **Audio:** An optional efficient streaming audio playback profile.
- **RFCOMM:** This block is responsible for RS-232 emulation and interfacing and is used for supporting telephony functionality.
- **Attribute protocol (ATT):** A wire application protocol used mainly in BLE (but can apply to BR/EDR). Optimized to run on BLE low power battery-based hardware. ATT is tightly coupled to GATT.
- **Generic Attribute Profile (GATT):** This block represents the functionality of the attribute server and, optionally, the attribute client. The profile describes the services used in the attribute server. Every BLE device must have a GATT profile. Principally, if not exclusively, used for BLE but could be used on a vanilla BR/EDR device.
- **Generic Access Profile (GAP):** Controls connections and advertising states. Allows a device to be visible to the outside world and forms the basis of all other profiles.
- AMP-specific stack:
 - **AMP (PHY):** Physical layer responsible for transmission and reception of data packets up to 24 Mbps.
 - **AMP MAC:** This is the media access control layer as defined in the IEEE 802 reference layer model. It provides address methods for the device.
 - **AMP PAL:** Layer interfacing the AMP MAC with the host system (L2CAP and AMP manager). This block translates commands from the host into specific MAC primitives and vice versa.
 - **AMP manager:** Uses L2CAP to communicate with a peer AMP manager on a remote device. Discovers remote AMP devices and determines their availability.

Bluetooth 5 PHY and interference

Bluetooth devices operate in the 2.4000 to 2.4835 GHz industrial, scientific, and medical (ISM) unlicensed frequency band. As mentioned earlier in this chapter, this particular unlicensed area is congested with a number of other wireless media, such as 802.11 Wi-Fi. To alleviate interference, Bluetooth supports **frequency-hopping spread spectrum (FHSS)**.



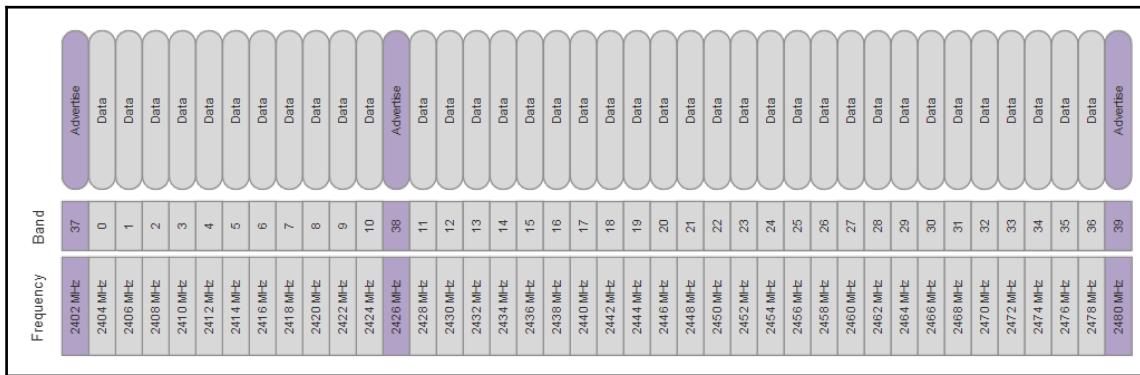
When having a choice between Bluetooth classic modes of BR/EDR, EDR will have less of a chance of interference and better coexistence with Wi-Fi and other Bluetooth devices since the on-air time is shorter due to its speed.

Adaptive Frequency Hopping (AFH) was introduced in Bluetooth 1.2. AFH uses two types of channels: used and unused. Used channels are in-play as part of the hopping sequence. Unused channels are replaced in the hopping sequence by used channels when needed in a random replacement method. BR/EDR mode has 79 channels and BLE has 40 channels. With 79 channels, BR/EDR mode has less than a 1.5 percent chance of interfering with another channel. This is what allows an office setting to have hundreds of headphones, peripherals, and devices all in the same range contending for frequency space (for example, fixed and continuous use sources of interference).

AFH allows a slave device to report channel classification information to a master to assist in configuring the channel hopping. In situations where there is interference with 802.11 Wi-Fi, AFH is used with a combination of proprietary techniques to prioritize traffic between the two networks. For example, if the hopping sequence regularly collides on channel 11, the master and slaves within the piconet will simply negotiate and hop over channel 11 in the future.

In BR/EDR mode, the physical channel is divided into slots. Data is positioned for transmission in precise slots and consecutive slots can be used if needed. By using this technique, Bluetooth achieves the effect of full-duplex communication through **Time Division Duplexing (TDD)**. BR uses **Gaussian Frequency-Shift Keying (GFSK)** modulation to achieve its 1 Mbps rate, while EDR uses **Differential Quaternary Phase Shift Keying (DQPSK)** modulation to 2 Mbps and 8-phase differential phase-shift keying (8DPSK) at 3 Mbps.

LE mode, on the other hand, uses **Frequency Division Multiple Access (FDMA)** and **Time Division Multiple Access (TDMA)** access schemes. With 40 channels rather than 79 for BR/EDR and each channel separated by 2 MHz, the system will divide the 40 channels into three for advertising and the remaining 37 for secondary advertising and data. Bluetooth channels are chosen pseudorandomly and are switched at a rate of 1600 hops/second. The following figure illustrates BLE frequency distribution and partitioning in the ISM 2.4 GHz space.



BLE frequencies partitioned into 40 unique bands with 2MHz separation. 3 Channels dedicated to advertising the remaining 37 dedicated to data transmission.

TDMA is used to orchestrate communication by requiring one device to transmit a packet at a predetermined time and the receiving device to respond at another predetermined time. The physical channels are subdivided into time units for particular LE events such as advertising, periodic advertising, extended advertising, and connecting. In LE, a master can form a link between multiple slaves. Likewise, a slave can have multiple physical links to more than one master and a device can be both a master and slave simultaneously. Role changes from master to slave or vice versa are not permitted.



As noted previously, 37 of the 40 channels are for data transmission but three are dedicated for advertising. Channels 37, 38 and 39 are dedicated to advertising GATT profiles. During advertising, a device will transmit the advertising packet on all three channels simultaneously. This helps increase the probability that a scanning host device will see the advertisement and respond.

Other forms of interference occur with mobile wireless standards in the 2.4 GHz space. Here, a technique called train nudging was introduced in Bluetooth 4.1.

Bluetooth 5.0 introduced **Slot Availability Masks (SAM)**. SAM allows two Bluetooth devices to indicate to each other the time slots that are available for transmission and reception. A map is built, indicating time slot availability. With the mapping, the Bluetooth controllers can refine their BR/EDR time slots and improve overall performance.

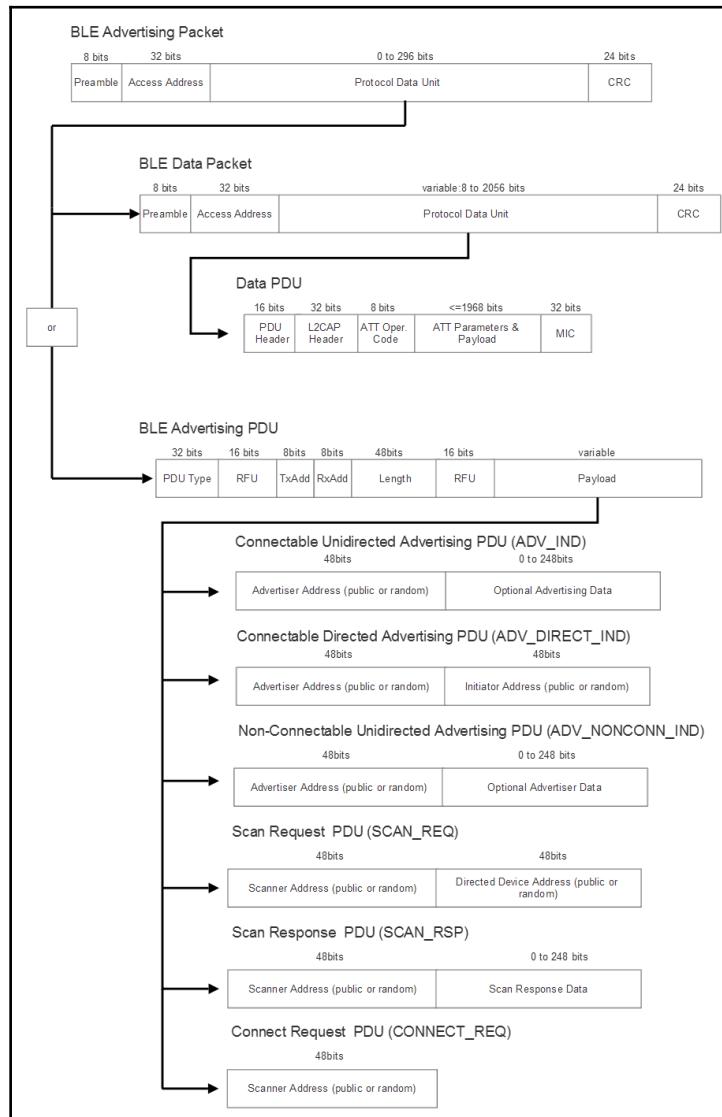
For BLE modes, SAM is not available. However, an overlooked mechanism in Bluetooth called **Channel Selection Algorithm 2 (CSA2)** can help with frequency hopping in noisy environments susceptible to multipath fading. CSA2 was introduced in Bluetooth 4.1 and is a very complex channel mapping and hopping algorithm. It improves the interference tolerance of the radio and allows the radio to limit the number of RF channels it can use in high interference locations. A side effect of limiting channels with CSA2 is it allows the transmit power to increase to +20dBm. As mentioned, there are limits to transmitting power imposed by governing regulatory bodies since BLE advertising channels and connected channels are so few. CSA2 allows for more channels to be available in Bluetooth 5 than in previous versions, which may open up regulatory restrictions.



Bluetooth packet structure

Every Bluetooth device has a unique 48-bit address called the BD_ADDR. The upper 24 bits of the BD_ADDR refer to the manufacturer specific address and are purchased through the IEEE Registration Authority. This address includes the **Organization Unique Identifier (OUI)**, also known as the company ID, and is assigned by the IEEE. The 24 least significant bits are free for the company to modify.

Three other secure and randomized address formats are available but will be discussed in the BLE security portion of this chapter. The following diagram shows the BLE advertising packet structure and various PDU types. This represents some of the most commonly used PDUs.



Common BLE Advertising and data packet formats. Several other packet types exist and should be referenced in the Bluetooth 5.0 Specification.

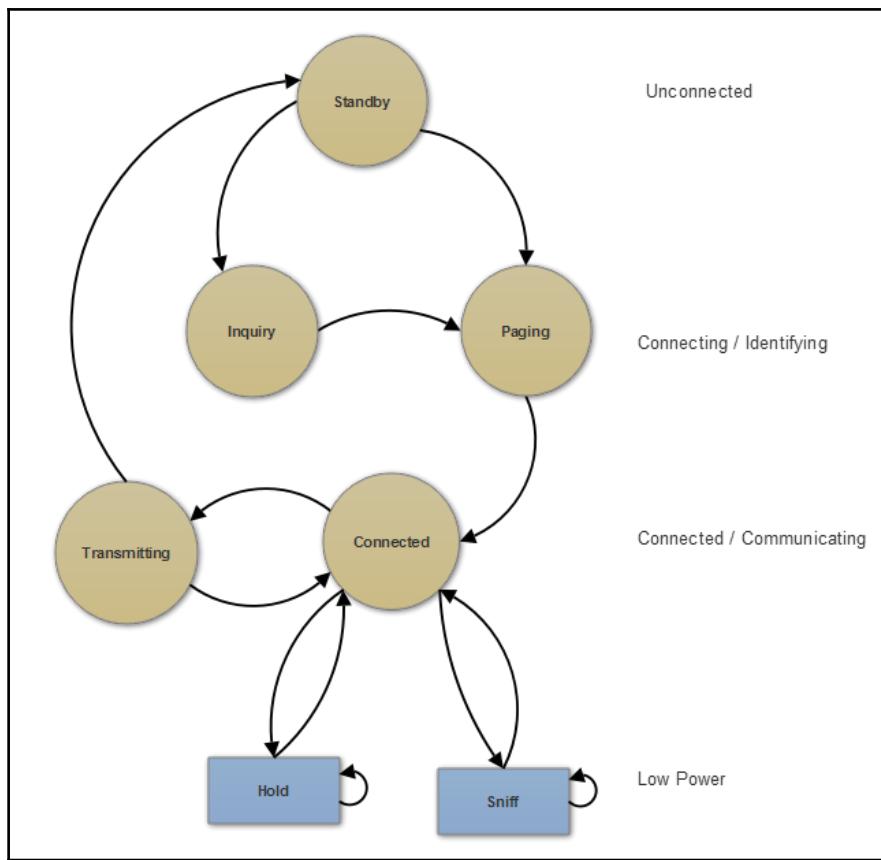
BR/EDR operation

Classic Bluetooth (BR/EDR) mode is connection-orientated. If a device is connected, the link is maintained even if no data is in being communicated. Before any Bluetooth connection transpires, a device must be discoverable for it to respond to scans of a physical channel and subsequently respond with its device address and other parameters. A device must be in a connectable mode to monitor its page scan.

The connection process proceeds in three steps:

1. **Inquiry:** In this phase, two Bluetooth devices have never associated or *bonded*; they know nothing of each other. The devices must discover each other through an inquiry request. If the other device is listening, it may respond with its BR_ADDR address.
2. **Paging:** Paging or connecting forms a connection between the two devices. Each device knows the BD_ADDR of each other at this point.
3. **Connected:** There are four sub-modes of the connection state. This is the normal state when two devices are actively communicating:
 1. **Active mode:** This is the normal mode of operation for transmission and receiving of Bluetooth data or waiting for the next transmission slot.
 2. **Sniff mode:** This is a power saving mode. The device is essentially asleep but will listen for transmissions during specific slots that can be changed programmatically (for example, 50 ms).
 3. **Hold mode:** This is a temporary low power mode initiated by the master or slave. It will not listen for transmissions like sniff mode, and the slave temporarily ignores ACL packets. While in this mode, switching to the connected state occurs very quickly.
 4. **Park mode:** As stated earlier, this mode is deprecated in Bluetooth 5.

A state diagram of these phases is as follows:



Connection process for Bluetooth from the unconnected device in standby mode, device query, and discovery, connected/transmitting mode, and low-power modes.

If this process completes successfully, the two devices can be forced to automatically connect when they are in range. The devices are now pairing. The one-time pairing process is most common in connecting smartphones to a vehicle stereo, but it can apply anywhere in the IoT as well. Paired devices will share a key that is used in the authentication process. More will be covered on keys and authentication in the security section of Bluetooth.



Apple recommends using a sniff mode set at a 15 ms interval. This saves significant power over keeping the device in an active mode, but also allows for better sharing of the spectrum with Wi-Fi and other Bluetooth signals in the area. Additionally, Apple recommends a device should first set an advertising interval for initial discovery by a host to 20 ms and broadcast that for 30 seconds. If the device still can't connect to the host, the advertising interval should be increased programmatically to increase the chance of completing the connection process. See *Bluetooth Accessory Design Guidelines for Apple Products Release 8*, Apple Computer, June 16, 2017.

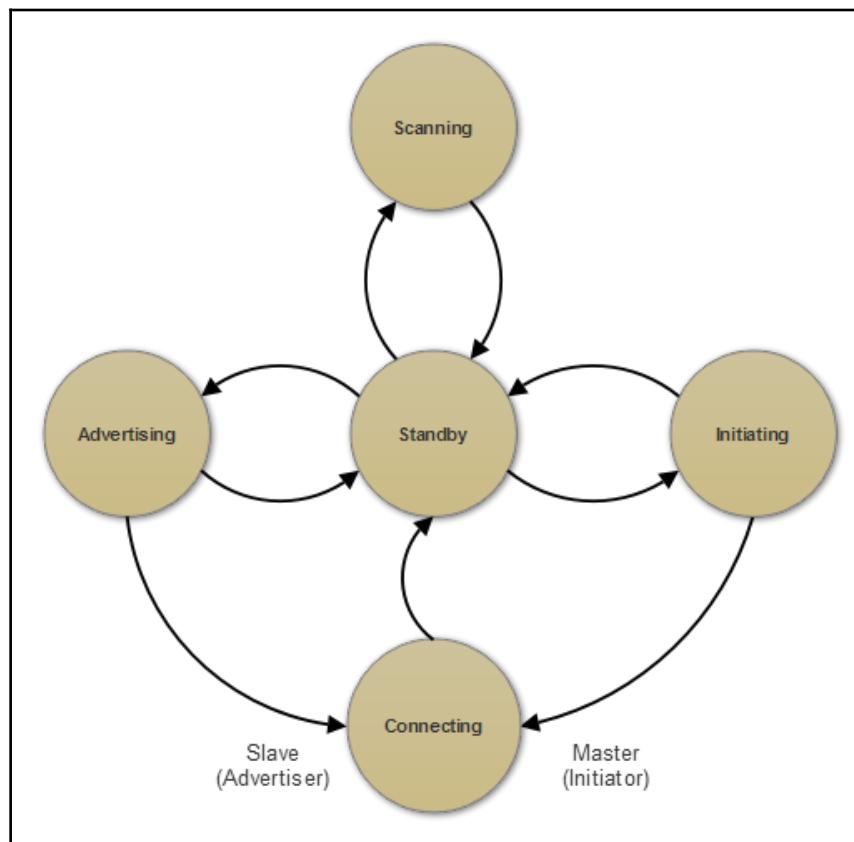
BLE operation

In Bluetooth Low Energy mode, there are five link states that are negotiated by the host and device:

- **Advertising:** Devices that transmit the advertising packets on the advertising channels.
- **Scanning:** Devices that receive advertising on the advertising channels without intention to connect. Scanning can be active or passive:
 - **Active scanning:** Link layer listens for advertising PDUs. Depending on the received PDU, it may request an advertiser to send additional information.
 - **Passive scanning:** Link layer will only receive packets, transmission disabled.
- **Initiating:** Devices that need to form a connection to another device listen for connectable advertising packets and initiate by sending a connect packet.
- **Connected:** There is a relationship between master and slave in the connected state. The master is the initiator and the slave is the advertiser:
 - **Central:** An initiator transforms the role and title to the central device.
 - **Peripheral:** Advertising device becomes the peripheral device.
- **Standby:** Device in the unconnected state.

The advertising state has several functions and properties. Advertisements can be general advertisements where a device broadcasts a general invitation to some other device on the network. A directed advertisement is unique and designed to invite a specific peer to connect as fast as possible. This advertisement mode contains the address of the advertising device and the invited device.

When the receiving device recognizes the packet, it will immediately send a connect request. The directed advertisement is to get fast and immediate attention and the advertisements are sent at a rate of 3.75 ms, but only for 1.28 seconds. A non-connectable advertisement is essentially a beacon (and may not even need a receiver). We will describe beacons later. Finally, the discoverable advertisement can respond to scan request, but it will not accept connections. Shown in the following state diagram are the five link states of BLE operation.



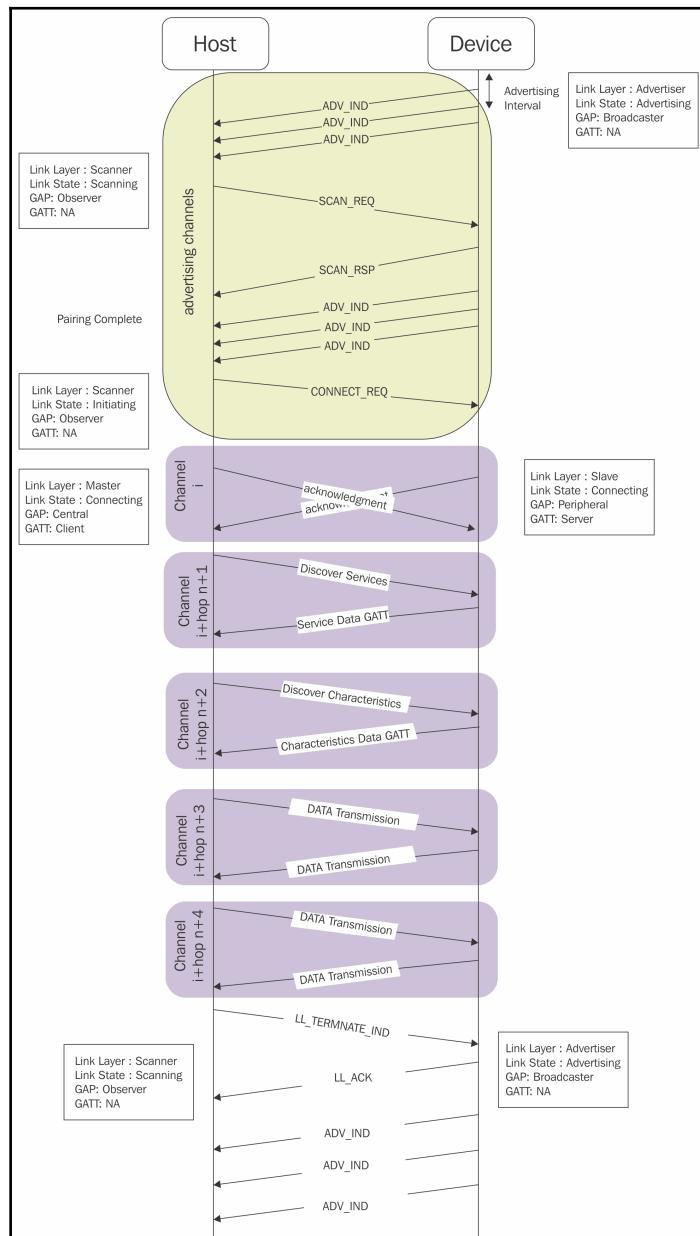
The BLE Link States.

A BLE device that has not previously bonded with a host initiates communication by broadcasting advertisements on the three advertising channels. The host can respond with a `SCAN_REQ` to request more information from the advertising device. The peripheral device responds with a `SCAN_RSP` and includes the device name or possibly services.



The `SCAN_RSP` can affect power usage on a peripheral device. If the device supports scan responses it must keep its radio active in receive mode, consuming power. This occurs even if no host device issues a `SCAN_REQ`. It is advisable to disable scan responses on IoT peripherals that are under power constraints.

After scanning, the host (scanner) initiates a `CONNECT_REQ`, at which point the scanner and advertiser will send empty PDU packets to indicate acknowledgment. The scanner is now termed the master and the advertiser is termed the slave. The master can discover slave profiles and services through the GATT. After discovery is complete, data can be exchanged from the slave to the master and vice versa. Upon termination, the master will return to a scanning mode and the slave will return to an advertiser mode. The following figure illustrates the BLE pairing process from advertising through data transmission.



Phases of BLE advertising, connecting, GATT service query and data transmission.

Bluetooth profiles

Applications interface with various Bluetooth devices using profiles. Profiles define the functionality and features for each layer of the Bluetooth stack. Essentially, profiles tie the stack together and define how layers interface with each other. A profile describes the discovery characteristics that the device advertises. They are also used to describe the data formats of services and characteristics that applications use to read and write to devices. A profile doesn't exist on a device; rather, they are predefined constructs maintained and governed by the Bluetooth SIG.

The basic Bluetooth profile must contain a GAP as stated by the specification. The GAP defines the radio, baseband layer, link manager, L2CAP, and service discovery for BR/EDR devices. Likewise, for a BLE device, the GAP will define the radio, link layer, L2CAP, security manager, attribute protocol, and generic attribute profile.

The ATT attribute protocol is a client-server wire protocol optimized for low-power devices (for example, the length is never transmitted over BLE, it is implied by the PDU size). ATT is also very generic and much is left to the GATT to provide assistance. The ATT profile consists of:

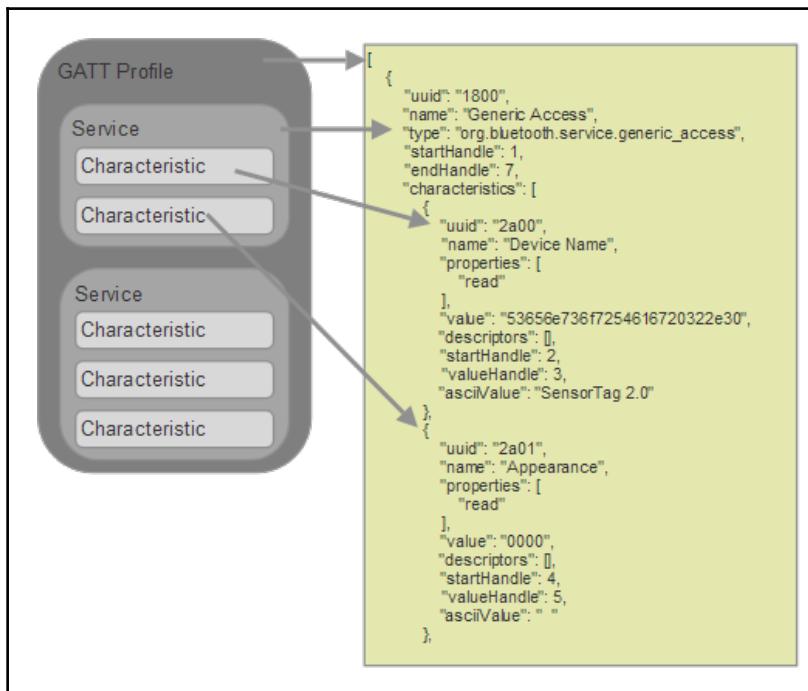
- A 16-bit handle
- A UUID to define the attribute type
- A value containing the length

The GATT logically resides on top of ATT and is used primarily, if not exclusively, for BLE devices. The GATT specifies the roles of the server and client. A GATT client is usually a peripheral device and the GATT server is the host (PC, smartphone). A GATT profile contains two components:

- **Services:** A service breaks up data into logical entities. There can be multiple services in a profile and each service has a unique UUID to distinguish it from others.

- **Characteristics:** A characteristic is the lowest level of a GATT profile and contains raw data associated with the device. Data formats are distinguished by 16-bit or 128-bit UUIDs. The designer is free to create their own characteristics that only their application can interpret.

The image below represents an example of a Bluetooth GATT profile with corresponding UUIDs for various services and characteristics.



GATT Profile Hierarchy and example GATT used on Texas Instruments CC2650 SensorTag

The Bluetooth SIG maintains a collection of many GATT profiles. At the time of writing, there are 57 GATT profiles supported on the Bluetooth SIG: <https://www.bluetooth.com/specifications/gatt>. Profiles supported by the SIG include everything from health monitors, cycling and fitness devices, environmental monitors, human interface devices, indoor positioning, object transfer, and location and navigation services, along with many others.

BR/EDR security

Bluetooth security has existed as part of the protocol since 1.0 in some form. We discuss security for BR/EDR mode and BLE separately as the mechanisms are different. Starting with BR/EDR mode, there are multiple modes of authenticating and pairing. For both BR/EDR and BLE security, it is recommended to read and follow the latest security guide provided by the US National Institute of Standards and Technology: *Guide to Bluetooth Security, NIST Special Publication (SP) 800-121 Rev. 2, NIST, 5/8/2017*.

Pairing requires the generation of a secret symmetric key. In BR/EDR mode, this is called the link key, while in BLE mode it's termed the *long-term key*. Older Bluetooth devices used a **Personal Identification Number (PIN)** pairing mode to initiate link keys. Newer devices (4.1+) use secure simple pairing.

Secure Simple Pairing (SSP) provides a pairing process with a number of different association models for various use cases. SSP also uses public key cryptography to protect from eavesdropping and **Man-in-the-Middle** attacks (MITM). The models supported by SSP include:

- **Numeric comparison:** For use cases where both Bluetooth devices can display a six-digit numeric value allowing a user to enter a yes/no response on each device if the numbers match.
- **Passkey entry:** Used in situations where one device has a numeric display and the other only has a numeric keyboard. In this case, the user enters the value seen on the first device's display on the second device's keyboard.
- **Just WorksTM:** For situations where one device is headless and has no keyboard or display. It only provides minimal authentication and will not thwart a MITM attack.
- **Out-of-band (OOB):** This is used when the device has a secondary form of communication such as NFC or Wi-Fi. The secondary channel is used for discovery and cryptographic value exchange. It will only protect from eavesdropping and MITM if the OOB channel is secure.

Authentication in BR/EDR mode is a challenge-response action; for example, entering a PIN on a keypad. If the authentication fails, the device will wait for an interval before allowing a new attempt. The interval grows exponentially with each failed attempt. This is simply to frustrate the individual attempting to manually break a key code.

Encryption in BR/EDR mode can be set so that it is disabled for all traffic, so that it is encrypted for data traffic but broadcast communication will be raw, or so that all communication is encrypted. The encryption uses AES-CCM cryptography.

BLE security

BLE pairing (explained earlier in this chapter) starts by a device initiating a *Pairing_Request* and exchanging capabilities, requirements, and so on. Nothing involving security profiles occurs at the initial phase of a pairing process. For that matter, the pairing security is similar to the BR/EDR four methods (also known as association models), but differs slightly in Bluetooth BLE 4.2:

- **Numeric comparison:** This is the same as Just Works, but at the end, both devices will generate a confirmation value that is displayed on the host and device screens for the user to validate the match.
- **Passkey entry:** Similar to BR/EDR mode, except that the non-initiating device creates a random 128-bit seed called a nonce to authenticate the connection. Every bit of the passkey is authenticated separately on each device by generating a confirmation value for that bit. The confirmation values are exchanged and should match. The process continues until all bits are processed. This provides a fairly robust solution for MITM attacks.
- **Just Works™:** After devices exchange public keys, the non-initiating device creates a nonce to generate a confirmation value C_b . It transmits the nonce and C_b to the initiating device, which in turn generates its own nonce and transmits it to the first. The initiating device will then confirm the authenticity of the non-initiating nonce by generating its own C_a value which should match C_b . If it fails to match, the connection is corrupt. This again is different to BR/EDR mode.
- **Out-of-band (OOB):** This is the same as in BR/EDR mode. It will only protect from eavesdropping and MITM if the OOB channel is secure.

In BLE, (as of Bluetooth 4.2) key generation uses LE secure connections. LE *secure connection* was developed to resolve the security hole in BLE pairing that allowed an eavesdropper to see the pairing exchange. This process uses a **long-term key (LTK)** to encrypt the connection. The key is based on an **elliptical-curve Diffie-Hellman (ECDH)** public key cryptography. Both the master and slave will generate ECDH public-private key pairs. The two devices will exchange their public portion of their respective pairs and process the Diffie-Hellman key. At this point, the connection can be encrypted using AES-CCM cryptography.

BLE also has the ability to randomize its `BD_ADDR`. Remember, the `BD_ADDR` is a 48-bit MAC-like address. Rather than a static address for the value, as mentioned earlier in this chapter, there are three other options:

- **Random static:** These addresses are either burned into the device's silicon during manufacture or generated when the device power cycles. If a device routinely power cycles, a unique address will be generated and this remains secure as long as the frequency of power cycling is high. In an IoT sensor environment, that may not be the case.
- **Random private resolvable:** This addressing method can only be used if the **Identity Resolving Key (IRK)** is exchanged between the two devices during the bonding process. The device will use the IRK to encode its address into a random address in the advertisement packet. A second device that also possesses the IRK will convert the random address back into the authentic address. In this method, the device will periodically generate a new random address based off the IRK.
- **Random private non-resolvable:** The device address is simply a random number and a new device address can be generated at any time. This offers the highest level of security.

Beaconing

Bluetooth beaconing is a secondary effect of BLE; however, it is an important and significant technology for IoT. Because beacons are not necessarily sensors, we didn't cover them explicitly in the *Chapter 3, Sensors, Endpoints, and Power Systems* (although some do provide sensing information in an advertisement packet). Beaconing is simply using Bluetooth devices in LE mode to advertise on some periodic basis. Beacons never connect or pair with a host. If a beacon were to connect, all advertisements would stop and no other device could hear that beacon. The three use cases important to retail, healthcare, asset tracking, logistics, and many other markets are:

- **Static point of Interest (POI)**
- Broadcasting telemetry data
- Indoor positioning and geolocation services

Bluetooth advertising uses a message to contain further information in the broadcast UUID. An app on a mobile device can respond to this advertisement and perform some action if the correct advertisement is received. A typical retail use case would make use of a mobile app that would respond to the presence of a beacon advertisement in the vicinity and pop up an ad or sale information on the user's mobile device. The mobile device would communicate over Wi-Fi or cellular to retrieve additional content as well as provide critical market and shopper data to the company.

Beacons can transmit their calibrated RSSI signal strength as an advertisement. The signal strength of beacons is calibrated by manufacturers typically at a one-meter length. Indoor navigation can be performed in three different ways:

- **Multiple beacons per room:** This is a simple triangulation method to determine a user's location-based on the advertised RSSI signal strength collected from numerous beacons in a room. Given the advertised calibrated level from each beacon and the received strength from each beacon, an algorithm could determine the approximate location of a receiver in a room. This assumes all beacons are in a fixed place.
- **One beacon per room:** In this scheme, one beacon is placed in each room, allowing a user to navigate between rooms and halls with the fidelity of the location being a room. This is useful in museums, airports, concert venues, and so on.
- **Few beacons per building:** In combination with accelerometers and gyros in a mobile device, multiple beacons in a building can allow for dead reckoning ability in a large open space. This allows a single beacon to set a starting location and the mobile device to reckon its location based on user movement.

There are two fundamental beaconing protocols used: Eddystone by Google and iBeacon by Apple. Legacy Bluetooth devices can only support a beacon message of 31 bytes. This restricts the amount of data a device can convey. Elaborate schemes have been used to help reduce the packet size and encode the message.

An entire iBeacon message is simply a UUID (16 bytes), a major number (two bytes), and minor number (two bytes). The UUID is specific to the application and use case. The major number further refines the use case and minor extends the use case to an even narrower case.

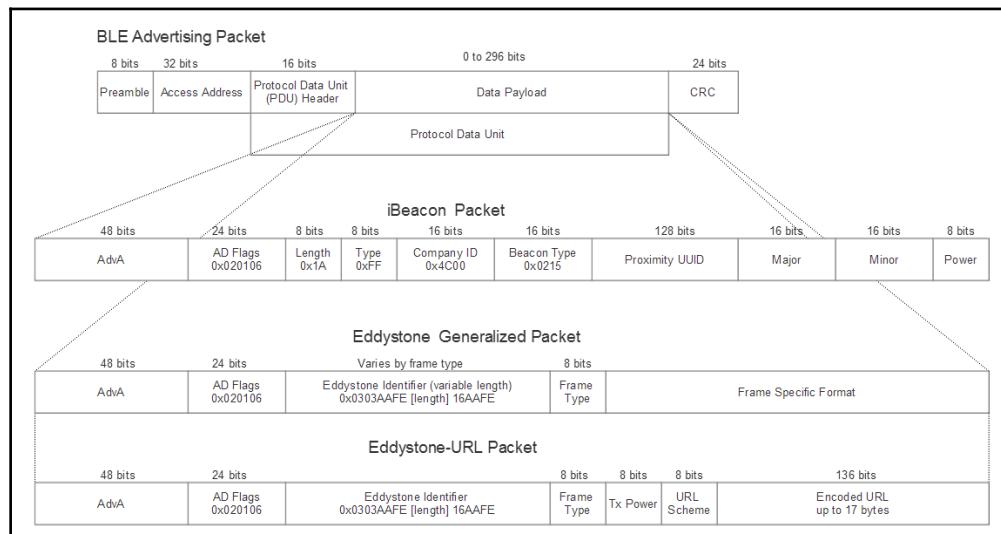
iBeacons provide two ways to detect devices:

- **Monitoring:** Monitoring works even if the associated smartphone application is not actively running
- **Ranging:** Ranging works only if the application is active

Eddystone (also known as UriBeacons) can transmit four different types of frames with varying length and frame encoding:

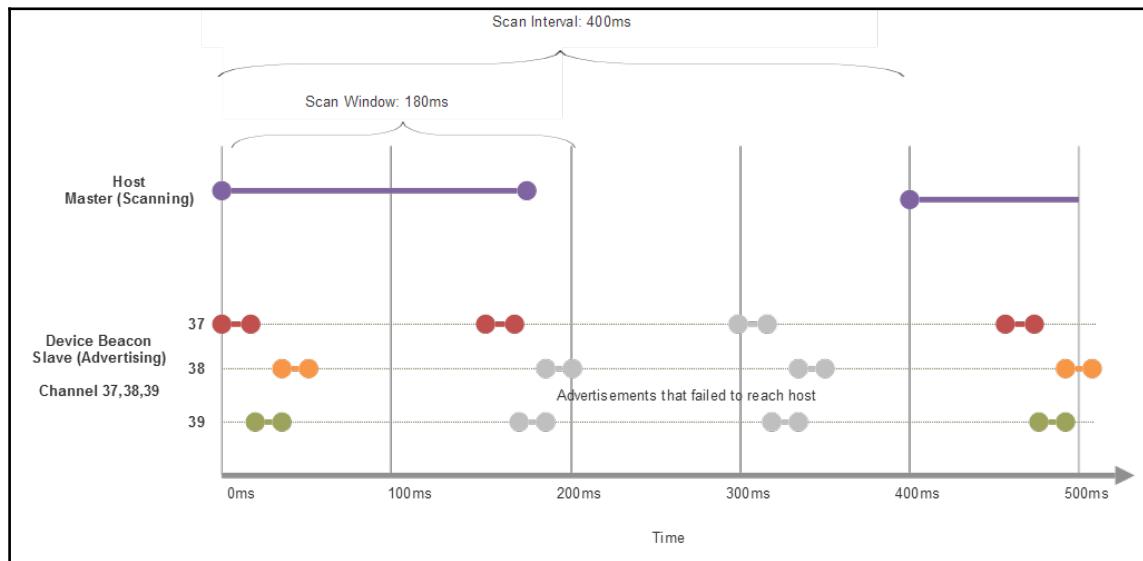
- **Eddystone-URL:** This is the uniform resource location. This frame allows a receiving device to display web content based on the location of the beacon. An app does not need to be installed to activate the content. The content is variable in length and applies unique compression schemes to reduce the size of the URL to a limit of 17 bytes.
- **Eddystone-UID:** 16-byte unique beacon ID with a 10-byte namespace and six-byte instance. Uses Google Beacon Registry to return attachments.
- **Eddystone-EID:** A short-lived identifier for beacons requiring higher levels of security. No fixed namespace and ID, the identifiers constantly rotate and require an authorized app to decode. Uses Google Beacon Registry to return attachments.
- **Eddystone-TLM:** Broadcasts telemetry data about the beacon itself (battery level, time since power-on, advertisement count). Broadcasts alongside URI or URL packets.

The following diagram illustrates the Bluetooth BLE advertisement packet structure for Eddystones and iBeacons. The iBeacon is simplest with a single type of frame of consistent length. Eddystone consists of four different types of frames and has variable lengths and encoding formats. Note that some fields are hard-coded such as the length, type, and company ID for iBeacon, as well as the identifier for Eddystone.



Example difference between iBeacon and Eddystone advertising packets (PDU).

The scanning intervals and advertising intervals attempt to minimize the number of advertisements necessary to convey useful data within a period of time. The scanning window is usually a longer duration than the advertisement as scanners essentially have more power than a coin cell battery in a beacon. The following figure shows a process of a beacon advertising every 180 ms while the host scans every 400 ms.

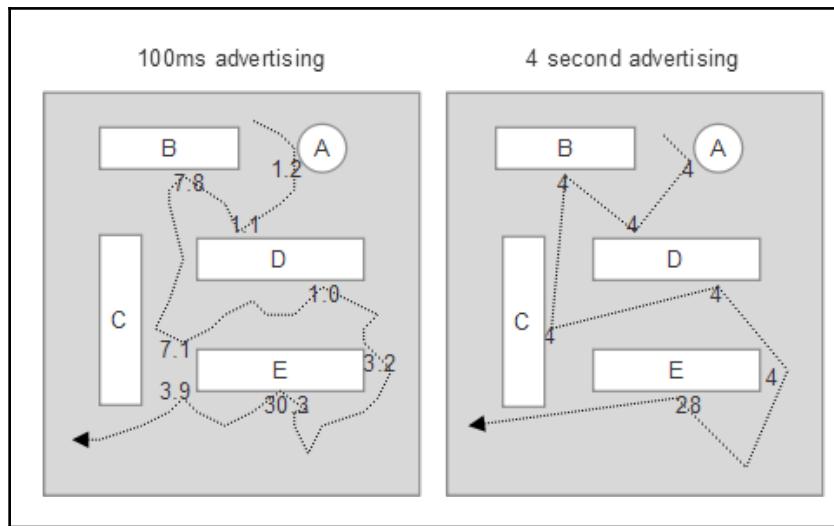


An example of a host scanning with a scan interval of 400 ms and a scan window of 180 ms.

The beacon advertises every 150ms on the dedicated channels 37,38,39. Notice the ordering of the advertising channels isn't consecutive as frequency hopping may have adjusted the order. Some advertisements have failed to reach the host since the scan interval and advertising interval is out of phase. Only a single advertisement in the second burst reach the host on channel 37, but by design, Bluetooth advertises on all three channels in an attempt to maximize the chance of success.

There are two fundamental challenges with architecting a beacon system. The first is the effect of advertising intervals versus the fidelity of location tracking. The second is the effect of advertising intervals versus the longevity of the battery powering the beacon. Both effects balance each other and careful architecture is needed to deploy correctly and extend the life of the battery.

The longer the interval between beacon advertisements, the less accuracy a system will have on a moving target. For example, if a retailer is tracking the location of patrons in a store and the patrons move at walking speed of 4.5 feet/second and a set of beacons advertises every four seconds versus another deployment that advertises every 100 ms, it will reveal different paths of motion to the retailer gathering market data. The following figure illustrates the effects of slow and fast advertising in a retail use case.



The effects of high-frequency advertising versus low-frequency advertising on location fidelity. The integers represent customer time spent at a particular point in the store.

A four-second advertising interval loses accuracy of the customer as they move about a store. Additionally, the amount of time spent at a particular point can only be tracked at discrete intervals of 4 seconds. The time spent at position B and C (as the customer leaves the store) may be lost. In this case, the retailer may want to understand why the customer spent 7.8 seconds at location B and why the customer went back to point C on their way out.

The counter effect of frequent advertising is the impact on beacon battery life. Typically, the batteries in beacons are Li-ion CR2032 coin cells. *The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs*. Web 14 March, 2016 (<https://www.aislelabs.com/reports/beacon-guide/>) has performed an analysis of battery life for some common beacons and altered their advertising interval (100 ms, 645 ms, and 900 ms). They also used different batteries with increasing stored energy. The results show that the average life ranges from 0.6 months to over one year depending on the chipset, but more importantly on the advertising interval. Along with the advertising interval, Tx power certainly affects the overall battery life, but so does the number of frames transmitted.



Setting advertising intervals too long, while beneficial for battery life and bad for location awareness, has a secondary effect. If the beacon is operating in a noisy environment and the interval is set high (>700 ms), the scanner (smartphone) will have to wait another full period to receive the advertisement packet. This can lead to apps timing out.

A fast 100 ms interval is useful to track fast-moving objects (such as asset tracking in fleet logistics or drone-based beacon harvesting). If the architect is designing to track human movement at a typical rate of 4.5 feet/second, then a 250 ms to 400 ms is adequate.

A valuable exercise for the IoT architect is to understand the cost of transmission in terms of power. Essentially, the IoT device has a limited number of PDUs it can send out before the battery reaches a point where it will not be able to power the device unless refreshed or energy is harvested (Chapter 4, *Communications and Information Theory*). Assume an iBeacon is advertised every 500 ms and the packet length is 31 bytes (it may be longer). Additionally, the device uses a CR2032 coin cell battery rated at 220mAh at 3.7V. The beacon electronics consume 49uA at 3V. We can now predict the life of the beacon and the efficiency of the transmission:

- $\text{Power consumption} = 49\mu\text{A} \times 3\text{V} = 0.147\text{mW}$
- $\text{Bytes per second} = 31 \times (1 \text{ second}/500 \text{ ms}) \times 3 \text{ channels} = 186 \text{ bytes/second}$
- $\text{Bits per second} = 186 \text{ bytes/second} \times 8 = 1488 \text{ bits/second}$
- $\text{Energy per bit} = 0.147 \text{ mW} / (1488 \text{ bits/second}) = 0.098 \text{ uJ/bit}$
- $\text{Energy used for each advertisement} = 0.098 \text{ uJ/bit} \times 31 \text{ bytes} \times 8 \text{ bits/byte} = 24.30 \text{ uJ/advertisement}$
- $\text{Energy stored in battery: } 220\text{mAh} \times 3.7\text{V} \times 3.6 \text{ seconds} = 2930 \text{ J}$
- $\text{Life of battery} = (2930 \text{ J} \times (1,000,000 \text{ uJ/J})) / ((24.30 \text{ uJ/advertisement}) \times (1 \text{ advertisement} / 0.5 \text{ seconds})) \times 0.7 = 42,201,646 \text{ seconds} = 488 \text{ days} = 1.3 \text{ years}$

The constant 0.7 is used for allowances in the decline of battery life, as described in Chapter 4, *Communications and Information Theory*. 1.3 years is a theoretical limit and most likely will not be obtained in the field due to factors such as leakage current as well as other functionalities of the device that may need to run periodically.

A final note on beacons with respect to Bluetooth 5. The new specification extends the length of beacon advertisements by allowing for advertisement packets to be transmitted in the data channels as well as the advertisement channels. This fundamentally breaks the limit of an advertisement being limited to 31 bytes.

With Bluetooth 5, the message size can be 255 bytes. The new Bluetooth 5 advertisement channels are called *secondary advertisement channels*. They ensure backward compatibility with legacy Bluetooth 4 devices by defining a specific Bluetooth 5 extended type in the header. Legacy hosts will throw out the unrecognized header and simply not listen to the device.

If a Bluetooth host receives a beacon advertisement that indicates there is a secondary advertisement channel, it recognizes that more data will need to be found in the data channels. The payload of the primary advertisement packet no longer contains beacon data, but a common extended advertising payload that identifies a data channel number and a time offset. The host will then read from that particular data channel at the indicated time offset to retrieve the actual beacon data. The data could also point to another packet (this is referred to as multiple secondary advertisement chains).

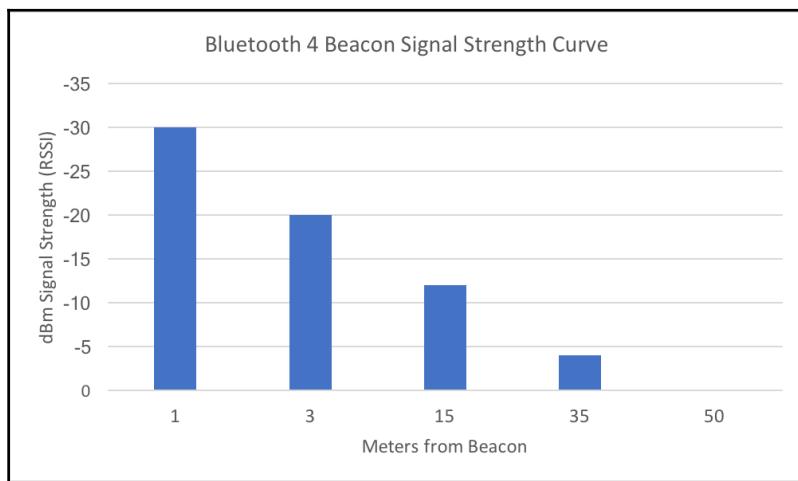
This new method of transmitting very long beacon messages ensures large amounts of data could be sent to customer smartphones. Other use cases and features are now enabled as well, such as the advertisement being used to transmit synchronous data such as audio streams. A beacon could send audio lectures to a smartphone as a visitor walks around a museum and looks at various art pieces.

Advertisements can also be anonymized, meaning the advertisement packet does not need to have the transmitter's address bound to it. Thus, when a device produces an anonymous advertisement, it does not transmit its own device address. This can improve privacy and reduce power consumption.

Bluetooth 5 can also transmit multiple individual advertisements (with unique data and different intervals) nearly simultaneously. This will allow Bluetooth 5 beacons to transmit Eddystone and iBeacon signals at nearly the same time without any reconfiguration. Additionally, the Bluetooth 5 beacon can detect if it is scanned by a host. This is powerful because the beacon can detect if a user received an advertisement and then stop transmitting to conserve power.

Bluetooth 5 range and speed enhancement

Bluetooth beacon strength is constrained and can be affected by limits placed on the transmitter power to preserve battery life. Usually, a line of sight is needed for optimal beacon range and signal strength. Shown below is the signal strength to distance curve for typical line-of-sight Bluetooth 4.0 communication.



Beacon strength is limited. Often a manufacturer limits the Tx power of a beacon to preserve battery life. As one moves from the beacon, the signal strength drops as expected.
Typically a 30-foot range is a usable beacon distance (Bluetooth 4.0)

Bluetooth also has different power levels, ranges, and transmission power based on a classification for each device:

Class number	Max output level (dBm)	Max output power (mW)	Max range	Use case
1	20 dBm	100 mW	100 m	USB adapter, access points
1.2	10 dBm	10 mW	30 m (typical 5 m)	Beacons, wearables
2	4 dBm	2.5 mW	10 m	Mobile devices, Bluetooth adapters, smart card reads
3	0 dBm	1 mW	10 cm	Bluetooth adapters

Bluetooth 5 has improved the range as well as data rate beyond legacy Bluetooth limits. A new radio PHY is available with Bluetooth 5 called LE2M. This doubles the raw data rate of Bluetooth from 1M symbol/second to 2M symbols/second. To transfer an equal amount of data on Bluetooth 5 as opposed to Bluetooth 4, less time will be needed for transmission. This is particularly relevant for IoT devices that operate on coin cells. The new PHY also increases the power from +10 dBm to +20 dBm, allowing for greater range.

With respect to range, Bluetooth 5 has another optional PHY for extended range transmission in BLE. This auxiliary PHY is labeled *LE coded*. This PHY still uses the 1M symbol/second rate as in Bluetooth 4.0, but reduces lower packet coding of 125 Kb/s or 500 Kb/s and increases the transmission power by +20dBm. This has the effect of increasing the range by 4x over Bluetooth 4.0 and having better penetration within buildings. The LE coded PHY does increase power consumption for the benefit of increased range.

Bluetooth mesh introduction

After the Bluetooth 5 specification was released the SIG focused on formalizing mesh networking in Bluetooth. The Bluetooth SIG published the mesh profile, device, and model specification 1.0 on July 13, 2017. This comes six months after the Bluetooth 5.0 specification was released. Before the official specification with Bluetooth 5 by the Bluetooth SIG, there had existed proprietary and ad hoc schemes using older Bluetooth versions to build mesh fabrics. The three specifications published by the Bluetooth SIG are as follows:

- **Mesh profile specification 1.0:** Defines the fundamental requirements to enable an interoperable mesh networking solution
- **Mesh model specification 1.0:** Basic functionality of nodes on the mesh network
- **Mesh device properties 1.0:** Defines the device properties required for the mesh model specification

It is not yet known if there is any limit to the size of the mesh network. There are some limits built into the specification. As of the 1.0 specification, there can be up to 32,767 nodes in a Bluetooth mesh and 16,384 physical groups. The maximum time-to-live, which is indicative of the depth of the mesh, is 127.



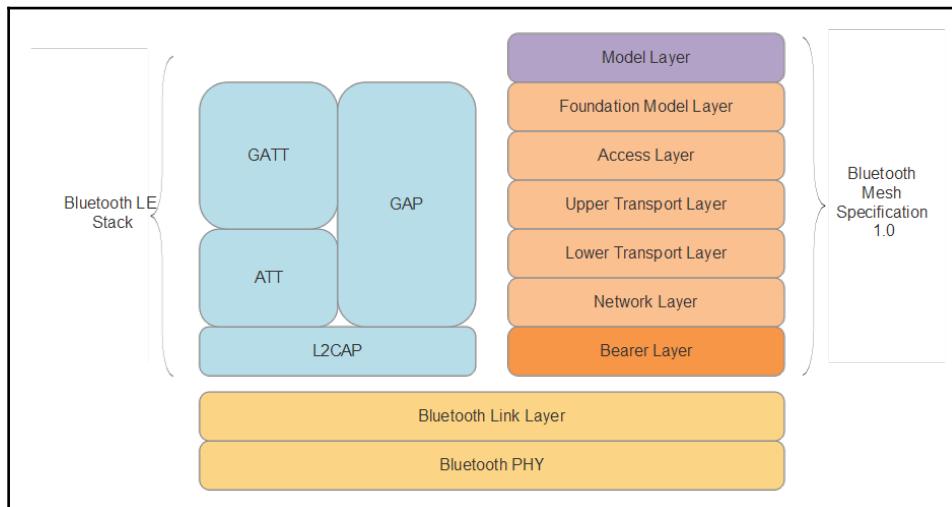
Bluetooth 5 mesh theoretically allows for 2^{128} virtual groups. Practically, grouping will be much more constrained.

Bluetooth mesh is based on BLE and sits on the BLE physical and link layer described earlier. On top of that layer is a stack of mesh-specific layers:

- **Models:** Implements behaviors, states, and bindings on one or more model specifications.
- **Foundation models:** Configuration and management of the mesh network.

- **Access layer:** Defines format of application data, encryption process, and data verification.
- **Upper transport layer:** Manages the authentication, encryption, and decryption of data passing to and from the access layer. Transports control messages such as friendships and heartbeats.
- **Lower transport layer:** Performs **segmentation and reassembly (SAR)** of fragmented PDUs if necessary.
- **Network layer:** Determines which network interface to output messages on. Manages the various address types and supports many bearers.
- **Bearer layer:** Defines how mesh PDUs are handled. Two types of PDUs are supported: advertising bearer and GATT bearer. The advertising bearer handles the transmission and reception of mesh PDUs, while the GATT bearer provides a proxy for devices that don't support the advertising bearer.
- **BLE:** The complete Bluetooth LE specification.

Bluetooth mesh can incorporate mesh networking or BLE functions. A device that is capable of mesh and BLE support can communicate to other devices like smartphones or have beacon functions. Shown below is the Bluetooth mesh stack. What is important to realize is the replacement of the stack above the link layer.

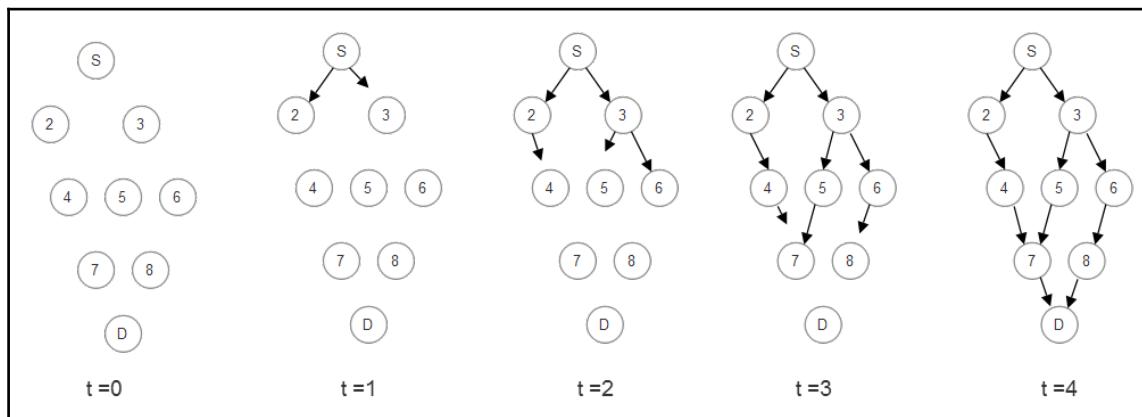


Bluetooth Mesh Specification 1.0 Stack

Bluetooth mesh topology

Bluetooth mesh uses the concept of a flood network. In a flood network, each incoming packet received by a node in the mesh is sent through every outgoing link, except the link to the parent of the message. Flooding has the advantage that if a packet can be delivered, it will be delivered (albeit probably many times via many routes). It will automatically find the shortest route (which can vary by signal quality and distance in a dynamic mesh). The algorithm is the simplest to implement as far as routing protocols are concerned.

Additionally, it requires no central manager such as a Wi-Fi network based around a central router. For comparison, the alternate types of mesh routing include tree-based algorithms. With tree algorithms (or cluster-tree algorithms), a coordinator is necessary to instantiate the network and becomes the parent node. Trees aren't necessarily true mesh networks, however. Other mesh routing protocols include *proactive routing*, which keeps up-to-date routing tables on each node, and *reactive routing*, which only updates routing tables on each node on demand; for example, when data needs to be sent through a node. Zigbee (covered later) is a form of proactive routing called an **Ad Hoc On-Demand Distance Vector (AODV)**. The following figure illustrates a flood broadcast. The time arrival at each level can vary dynamically from node to node. Additionally, the mesh network must be resilient to duplicate messages arriving at any node as in the case of node 7 and node D.



Flood Mesh Architecture. S= Source, D=Destination. Source produces data which propagates and flows through every node in the mesh.

The main disadvantage with a flood network is bandwidth waste. Depending on the fan out from each node, the congestion on a Bluetooth mesh can be significant. Another issue is a denial-of-service attack. If the mesh simply fans out messages, a facility is needed to know when to stop transmissions. Bluetooth accomplishes this through time-to-live identifiers, which we will cover later.

The nodes in a Bluetooth mesh include the following:

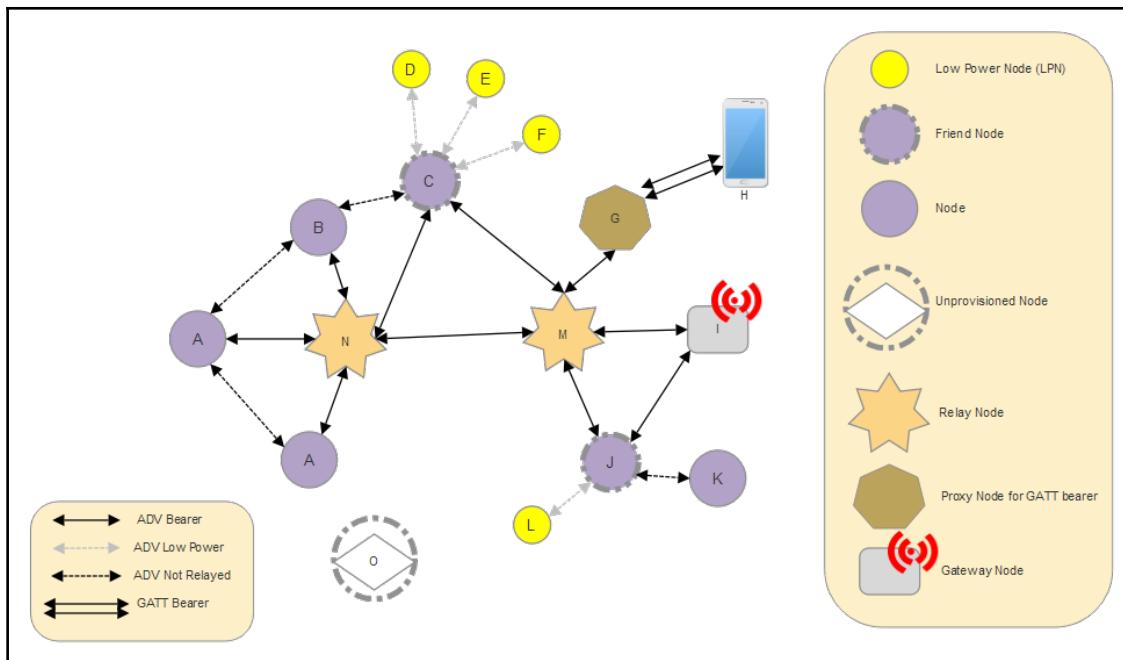
- **Nodes:** These are Bluetooth devices that have been previously provisioned and are members of a mesh.
- **Unprovisioned devices:** These are devices with the potential to join a mesh fabric that is not yet part of a mesh and has not been provisioned.
- **Elements:** A node with multiple constituent parts. Each part can be independently controlled and addressed. An example could be a Bluetooth node with temperature, humidity, and lumens sensors. This would be a single node (sensor) with three elements.
- **Mesh gateway:** A node that can translate messages between the mesh and a non-Bluetooth technology.

Once provisioned, a node may support an optional set of features, which include:

- **Relay:** A node that supports relay is termed a *relay node* and can retransmit messages received.
- **Proxy:** Allows for Bluetooth LE devices that do not support Bluetooth mesh natively to interact with nodes on the mesh. This is performed using a proxy node. The proxy exposes a GATT interface with the legacy Bluetooth device, and a proxy protocol based on a connection-oriented bearer is defined. The legacy device reads and writes to the GATT proxy protocol and the proxy node transforms the messages into true mesh PDUs.
- **Low power:** Some nodes on the mesh need to obtain extremely low levels of power consumption. They may serve up environmental sensor information (such as temperature) once an hour and be configured by a host or cloud managed tool once a year. That type of device cannot be placed in a listening mode when a message will only arrive once a year. The node enters a role termed the **low power node (LPN)**, which pairs it with a friend node. The LPN enters a deep sleep state and polls the associated friend for any messages that may have arrived while it was sleeping.

- **Friend:** The friend node is associated with the LPN but is not necessarily power constrained like an LPN. A friend may use a dedicated circuit or wall power. The friend's duty is to store and buffer messages destined for the LPN until the LPN wakes and polls it for messages. Many messages may be stored and the friend will transmit them in order using a **more data (MD)** flag.

The following diagram illustrates a Bluetooth mesh topology with the various components as they would be associated with one another in a real mesh.



Bluetooth Mesh Topology. Note the classes including Nodes, LPNs, Friends, Gateways, Relay Nodes, and Unprovisioned Nodes.

A Bluetooth mesh will cache messages on each node; this is crucial in a flood network. As the same message may arrive at different times from different sources, the cache provides a lookup of recent messages received and processed. If the new message is identical to one in the cache, it is discarded. This ensures system idempotence.

Each message carries a **time-to-live field (TTL)**. If a message is received by a node and then re-transmitted, the TTL is decremented by one. This is a safety mechanism to prevent endless loops, and networks create amplifying denial-of-service attacks internal to the mesh.

A heartbeat message is broadcast periodically from each node to the mesh. The heartbeat informs the fabric that the node is still present and healthy. It also allows the mesh to know how far away the node is and if it has changed distance since the last heartbeat. Essentially, it is counting the number of hops to reach the node. This process allows the mesh to re-organize and self-heal.

Bluetooth mesh addressing modes

Bluetooth mesh uses three forms of addressing:

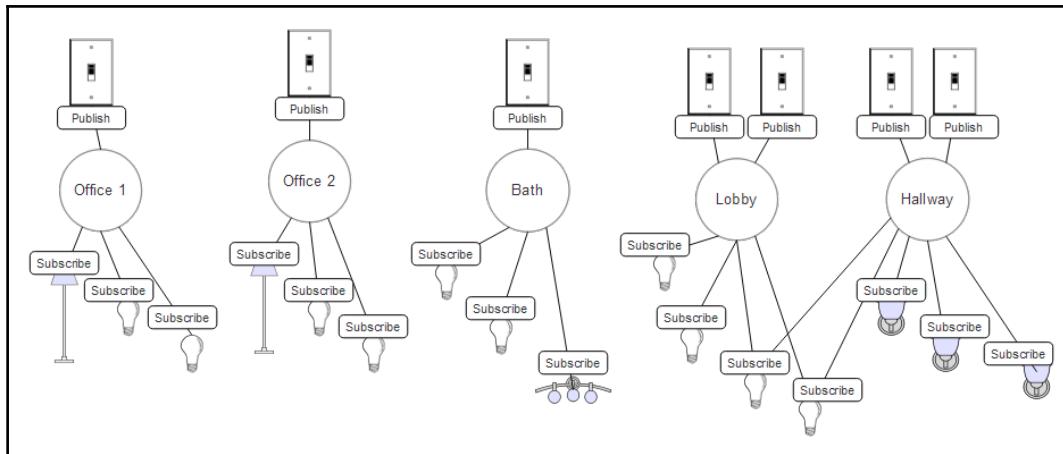
- **Unicast addressing:** Uniquely identifies a single element in the mesh. The address is assigned during the provisioning process.
- **Group addressing:** This is a form of multicast addressing that can represent one or more elements. These are either predefined by the Bluetooth SID as SIG fixed group addresses or assigned on the fly.
- **Virtual addressing:** An address can be assigned to more than one node and more than one element. Virtual addressing uses a 128-bit UUID. The intent is the UUID can be preset by a manufacturer to allow them to address their product globally.

The Bluetooth mesh protocol starts with 384-byte long messages that are segmented into 11-byte parcels. All communication in a Bluetooth mesh is message-orientated. There are two forms of messages that can be transmitted:

- **Acknowledged messages:** These require a response from node(s) that receive the message. The acknowledgment also includes data that the originator requested in the original message. Therefore, this acknowledged message serves dual purposes.
- **Unacknowledged messages:** These do not require a response from the receiver.

To send a message from a node is also known as publishing. Nodes configured to process select messages sent to particular addresses is termed subscribing. Each message is encrypted and authenticated using a network key and an application key.

The application key is particular to an app or use case (for example, turning a light on versus configuring the color of an LED light). Nodes will publish events (light switches) other nodes will subscribe to those events (lamps and lightbulbs). The following figure illustrates a Bluetooth mesh topology. Here, nodes can subscribe to multiple events (lobby lights and hallway lights). The circles represent group addresses. A switch would publish to a group.



Bluetooth Mesh Publish-Subscribe model.

Bluetooth mesh introduces the concept of group messaging. In a mesh, you may have a grouping of similar objects such as bath lights or lobby lights. This aids in usability; for example, if a new light is added, only that light needs to be provisioned, and the rest of the mesh doesn't need any change.

The light switches in the preceding example have two states: on and off. Bluetooth mesh defines states and in this case, they are labeled generic OnOff. Generic states and messages support many types of devices from lights to fans to actuators. They are a quick way to reuse a model for a general (or generic) purpose. As the system moves from one state to another, this is termed a state transition on the mesh. States can also be bound to each other. If a state changes, it can affect the transition to another state. As an example, a mesh controlling a ceiling fan may have a speed control state which, when it gets to a value of zero, changes the generic On-Off state to off.

Properties are similar to states but have more than binary values. For example, a temperature sensor may have a state Temperature 8 to signify and publish an eight-bit temperature value. A property can either be set as manufacturer (read-only) by the supplier of the element or admin, which allows read-write access. Both states and properties are communicated through messages on the mesh. Messages come in three types:

- **Get:** Requests a value of a given state from one or more nodes
- **Set:** Changes a value of a state
- **Status:** Is a response to a get that contains the data

All of these concepts of states and properties ultimately form a model (the highest level of the Bluetooth mesh stack). A model can be a server, in which case it defines states and state transitions. Alternatively, a model can be a client which doesn't define any states; rather, it defines state interaction messages to use with get, set, and status. A control model can support a hybrid of server and client models.

Mesh networking can also make use of Bluetooth 5 standard features such as anonymous advertisements and multiple advertisement sets. Take the case of a mesh that connects to a phone for voice communication, yet simultaneously is relaying packets for other uses. By using multiple advertisement sets, both use cases can be managed simultaneously.

Bluetooth mesh provisioning

A node may join a mesh by the act of provisioning. Provisioning is a secure process that takes an unprovisioned and insecure device and transforms it into a node in the mesh. The node will first secure a NetKey from the mesh. At least one NetKey must be on each device in the mesh to join. Devices are added to the mesh through a provisioner. The provisioner distributes the network key and a unique address to an unprovisioned device. The provisioning process uses elliptical-curve Diffie-Hellman key exchange to create a temporary key to encrypt the network key. This provides security from a man-in-the-middle attack during provisioning. The device key derived from the elliptical curve is used to encrypt messages sent from the provisioner to the device.

The provisioning process is as follows:

1. An unprovisioned device broadcasts a mesh beacon advertising packet.
2. The provisioner sends an invitation to the device. The unprovisioned device responds with a provisioning capabilities PDU.

3. The provisioner and device exchange public keys.
4. The unprovisioned device outputs a random number to the user. The user enters the digits (or identity) into the provisioner and a cryptographic exchange starts to complete the authentication phase.
5. A session key is derived by each of the two devices from the private key and the exchanged public keys. The session key is used to secure the data needed to complete the provisioning process, including securing the NetKey.
6. The device changes state from an unprovisioned device to a node and is now in possession of the NetKey, a unicast address, and a mesh security parameter called the IV index.

IEEE 802.15.4

The IEEE 802.15.4 is a standard wireless personal area network defined by the IEEE 802.15 working group. The model was ratified in 2003 and forms the basis of many other protocols including Thread (covered later), Zigbee (covered later), WirelessHART, and others.

802.15.4 only defines the bottom portion (PHY and data link layer) of the stack and not the upper layers. It is up to other consortiums and working groups to build a full network solution. The goal of 802.15.4 and the protocols that sit on it are low-cost WPAN with low power consumption. The latest specification is the IEEE 802.15.4e specification ratified on February 6, 2012, which is the version we will discuss in this chapter.

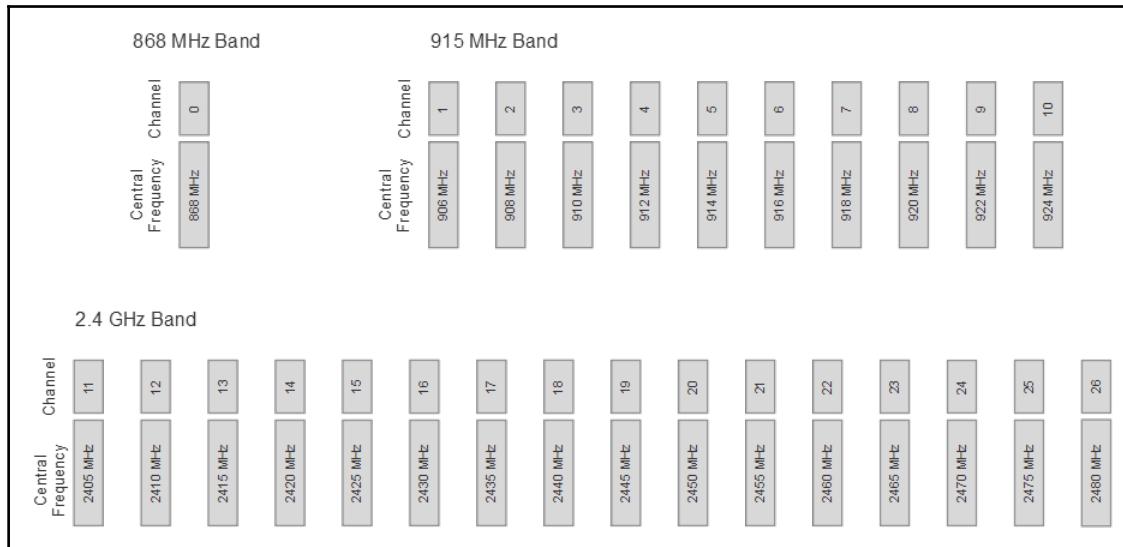
IEEE 802.15.4 architecture

The IEEE 802.15.4 protocol operates in the unlicensed spectrum in three different radio frequency bands: 868 MHz, 915 MHz, and 2400 MHz. The intent is to have as wide a geographical footprint as possible, which implies three different bands and multiple modulation techniques. While the lower frequencies allow 802.15 to have fewer issues with RF interference or range, the 2.4 GHz band is by far the most often used 802.15.4 band worldwide. The higher frequency band has gained its popularity because the higher speed allows for shorter duty cycles on transmitting and receiving, thus conserving power.

The other factor that has made the 2.4 GHz band popular is its acceptance in the market due to the popularity of Bluetooth. The table below lists various modulation techniques, geographical area, and data rates for various 802.15.4 bands.

Frequency range (MHz)	Channel numbers	Modulation	Data rate (Kbps)	Region
868.3	1 channel: 0	BPSK O-QPSK ASK	20 100 250	Europe
902-928	10 channels: 1-10	BPSK O-QPSK ASK	40 250 250	North America, Australia
2405-2480	16 channels: 11-26	O-QPSK	250	Worldwide

The typical range of an 802.15.4-based protocol is roughly 200 meters in an open-air, line-of-sight test. Indoors, the typical range is roughly 30 m. Higher power transceivers (15 dBm) or mesh networking can be used to extend the range. The following graphic shows the three bands used by 802.15.4 and the frequency distribution.



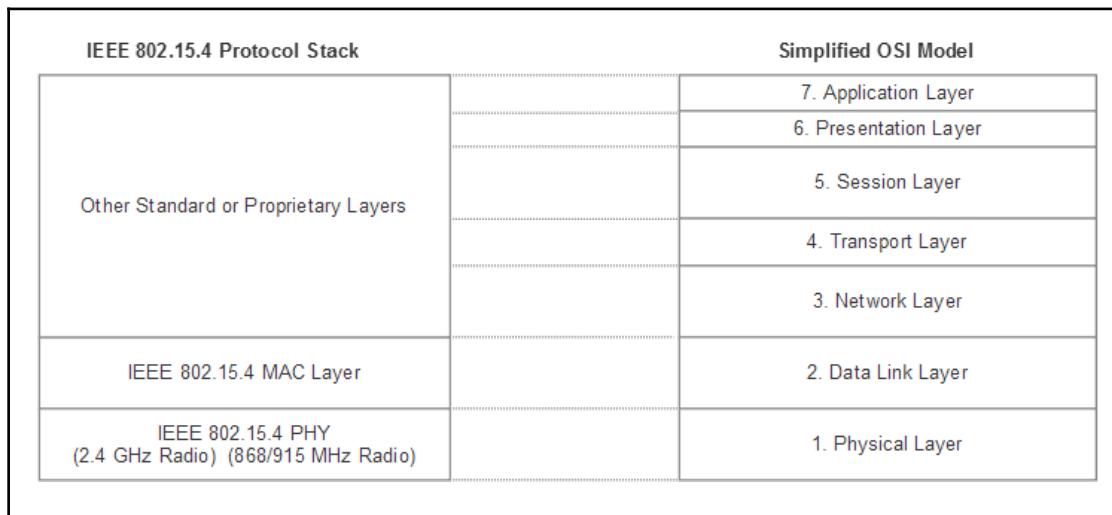
IEEE 802.15.4 bands and frequency allocations. 915 MHz band uses a 2MHz frequency separation and the 2.4GHz band uses a 5 MHz frequency separation.

To manage a shared frequency space, 802.15.4 and most other wireless protocols use some form of **Carrier Sense Multiple Access Collision Avoidance (CSMA/CA)**. Since it is impossible to listen to a channel while transmitting on the same channel, collision detection schemes don't work; therefore, we use collision avoidance. CSMA/CA simply listens to a specific channel for a predetermined amount of time. If the channel is sensed "idle", then it transmits by first sending a signal telling all other transmitters the channel is busy. If the channel is busy, then the transmission is deferred for a random period of time. In a closed environment, CSMA/CA will provide 36 percent channel usage; however, in real-world scenarios, only 18 percent of the channels will be usable.

The IEEE 802.15.4 group defines the operational transmit power to be capable of at least 3 dBm and the receiver sensitivity to be -85 dBm at 2.4 GHz and -91 dBm at 868/915 MHz. Typically, this implies a 15 mA to 30 mA current for transmission and a 18mA to 37mA current for the reception.

The data rate as state peaks at 250 kbps, as stated using the offset quadrature phase shift key.

The protocol stack only consists of the bottom two layers of the OSI model (PHY and MAC). The PHY is responsible for symbol encoding, bit modulation, bit demodulation, and packet synchronization. It also performs transmit-receiving mode switching and intra-packet timing/acknowledgment delay control. Shown below is the 802.15.4 protocol stack as a comparison to the OSI model.



IEEE 802.15.4 Protocol Stack. Only PHY and MAC layers are defined, other standards and organizations are free to incorporate layers 3 to 7 above the PHY and MAC.

On top of the physical layer is the data link layer responsible for detecting and correcting errors on the physical link. This layer also controls the media access layer (MAC) to handle collision avoidance using protocols such as CSMA/CA. The MAC layer is typically implemented in software and runs on an MCU such as the popular ARM Cortex M3 or even an 8-bit ATmega core. There are a few silicon vendors that have incorporated the MAC in pure silicon, such as Microchip Technology Inc.

The interface from the MAC to the upper layers of the stack are provided through two interfaces called the **Service Access Points (SAP)**:

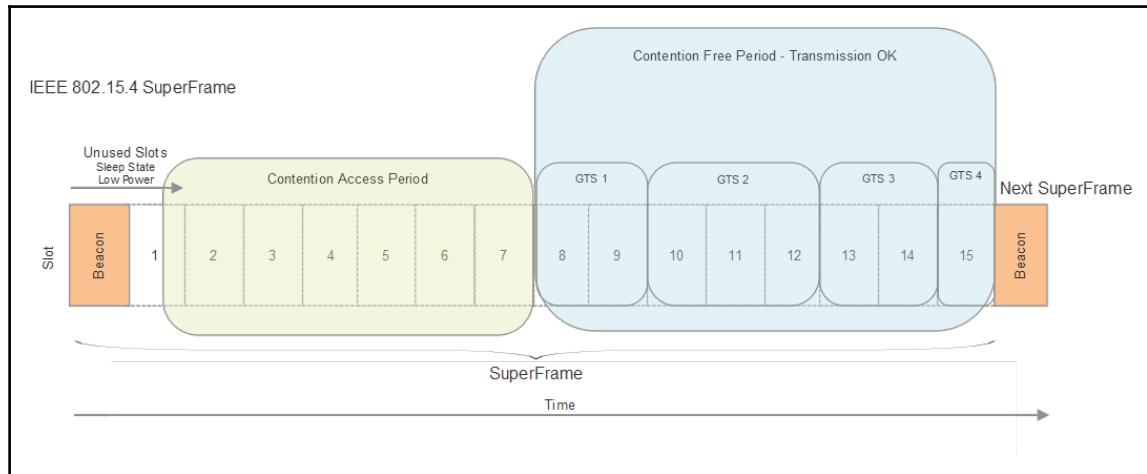
- **MAC-SAP:** For data management
- **MLME-SAP:** For control and monitoring (MAC layer management entity)

There are two types of communication in IEEE 802.15.4: beacon and beaconless communication.

For a beacon-based network, the MAC layer can generate beacons that allow a device to enter a PAN as well as provide timing events for a device to enter a channel to communicate. The beacon is also used for battery-based devices that are normally sleeping. The device wakes on a periodic timer and listens for a beacon from its neighbors. If a beacon is heard, it begins a phase called a SuperFrame Interval where time slots are pre-allocated to guarantee bandwidth to devices, and devices can call for a neighbor node attention. The **SuperFrame Interval (SO)** and **Beacon Interval (BO)** are fully controllable by the PAN coordinator. The SuperFrame is divided into sixteen equally sized time slots with one dedicated as the beacon of that SuperFrame. Slotted CSMA/CA channel access is used in beacon-based networks.

The guaranteed time slots (GTS) can be assigned to specific devices preventing any form of contention. Up to seven GTS domains are allowed. The GTS slots are allocated by the PAN coordinator and announced in the beacon it broadcasts. The PAN coordinator can change the GTS allocations on the fly dynamically based on system load, requirements, and capacity. The GTS direction (transmit or receive) is predetermined before the GTS starts. A device may request one transmit and/or one receive GTS.

The SuperFrame has **contention access periods (CAP)** where there is crosstalk on the channel and **contention free periods (CFP)** where the frame can be used for transmission and GTS. The following figure illustrates a superframe consisting of 16 equal time slots bounded by beacon signals (one of which must be a beacon). Contention Free Period will be further divided into Guaranteed Time Slots (GTS) and one or more GTSW may be allocated to a particular device. No other device may use that channel during a GTS.



IEEE 802.15.4 SuperFrame Sequence.

In addition to beacon-based networking, IEEE 802.15.4 allows for beacon-less networking. This is a much simpler scheme where no beacon frame is transmitted by the PAN coordinator. It implies, however, that all nodes are in a receiving mode all the time. This provides full-time contention access through the use of unslotted CSMA/CA. A transmitting node will perform a **clear channel assessment (CCA)** in which it listens to the channel to detect if it's used and then transmit if clear. CCA is part of a CSMA/CA algorithm and is used to "sense" if a channel is used. A device can receive access to a channel if it is clear of other traffic from other devices (including non-802.15.4 devices). In the event a channel is busy, the algorithm enters a "back-off" algorithm and waits a random amount of time to retry the CCA. The IEEE 802.15.4 group specifies the following for CCA usage:

- **CCA mode 1:** Energy above threshold (lowest). The CCA will report a busy medium on detecting any energy greater than a threshold value (ED).
- **CCA mode 2:** Carrier sense only (medium - default). This mode causes CCA to report a busy medium only if a **direct-sequence spread spectrum (DSSS)** signal is detected. The signal may be above or below the ED threshold.
- **CCA mode 3:** Carrier sense with energy above threshold (strongest). In this mode, the CCA reports busy if it detects a DSSS signal with an energy above the ED threshold.

- **CCA mode 4:** This mode is a carrier sense detection mode with a timer. The CCA starts a timer of some number of milliseconds and will report busy only if it detects a high-rate PHY signal. CCA will report the medium is idle if the timer expires and no high-rate signal is observed.
- **CCA Mode 5:** This is a combination of carrier sense and energy above a threshold mode.

To note regarding CCA modes:

- Energy detection will be at most 10 dB above specified receiver sensitivity
- CCA detection time will equal eight symbol periods
- It should be noted that CCA modes that are based on energy detection consume the least amount of energy for the device.

This mode will consume much more power than beacon-based communication.

IEEE 802.15.4 topology

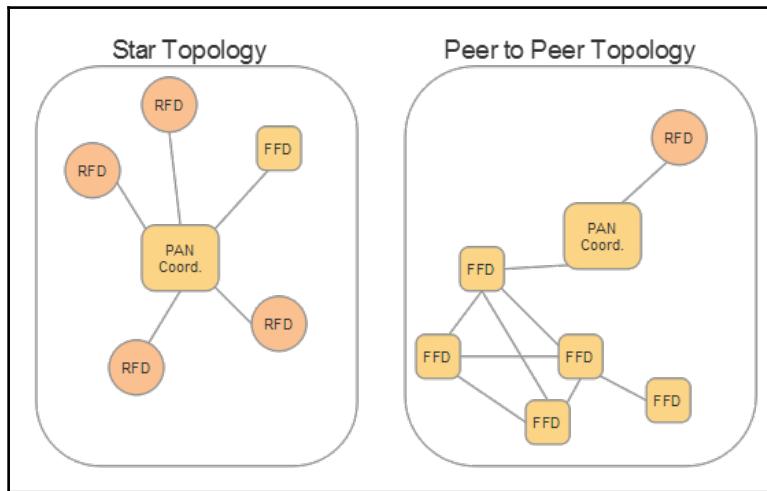
There are two fundamental device types in IEEE 802.15.4:

- **Full function device (FFD):** Supports any network topology, can be a network (PAN) coordinator and can communicate to any device PAN coordinator
- **Reduced function device (RFD):** Limited to only a star topology, cannot perform as a network coordinator, can only communicate with a network coordinator

The star topology is the simplest but requires all messages between peer nodes to travel through the PAN coordinator for routing. A peer-to-peer topology is a typical mesh and can communicate directly with neighbor nodes. To build more complicated networks and topologies is the duty of the higher level protocols, which we will discuss in the *Zigbee* section.

The PAN coordinator has a unique role that is to set up and manage the PAN. It also has the duty of transmitting network beacons and storing node information. Unlike sensors that may use battery or energy harvesting power sources, the PAN coordinator is constantly receiving transmissions and is usually on a dedicated power line (wall power). The PAN coordinator is always an FFD.

The RFD or even low power FFDs can be battery based. Their role is to search for available networks and transfer data as necessary. These devices can be put into a sleep state for very long periods of time. Shown below is a diagram of a star versus peer-to-peer topology.



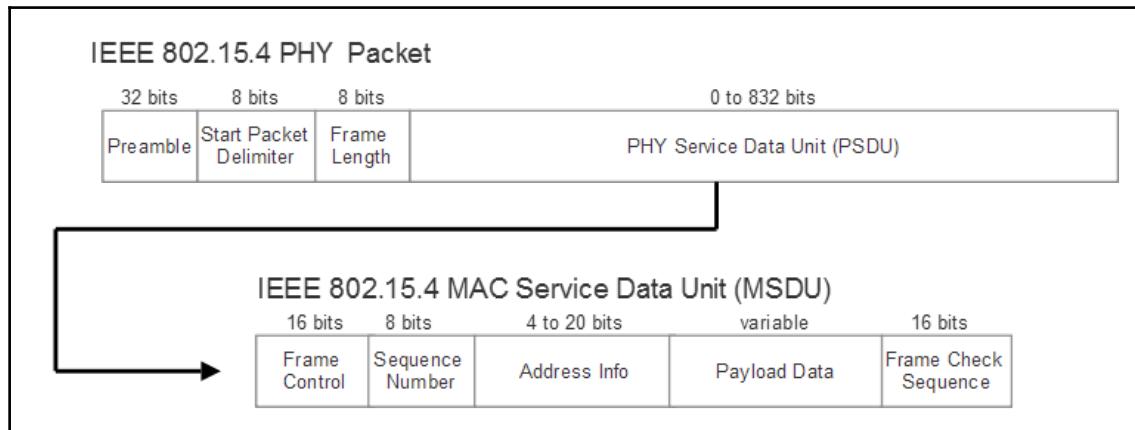
IEEE 802.15.4 guidance on network topologies. Implementer of 802.15.4 is free to build other network topologies

Within the PAN, broadcast messages are allowed. To broadcast to the entire fabric, one only needs to specify the PAN ID of 0xFFFF.

IEEE 802.15.4 address modes and packet structure

The standard dictates that all addresses are based on unique 64-bit values (IEEE address or MAC address). However, to conserve bandwidth and reduce the energy of transmitting such large addresses, 802.15.4 allows a device joining a network to "trade-in" their unique 64-bit address for a short 16-bit local address, allowing for more efficient transmission and lower energy.

This "trade-in" process is the responsibility of the PAN coordinator. We call this 16-bit local address the PAN ID. The whole PAN network itself has a PAN identifier, as multiple PANs can exist. The following figure is a diagram of the 802.15.4 packet structure.



IEEE 802.15.4 PHY and MAC packet encoding.

Frames are the basic unit of data transport, and there are four fundamental types (some of the underlying concepts were covered in the last section):

- **Data frame:** Application data transport
- **Acknowledgement frame:** Confirmation of reception
- **Beacon frame:** Sent by PAN coordinator to set up a SuperFrame structure
- **MAC command frame:** MAC layer management (associate, disassociate, beacon request, GTS request)

IEEE 802.15.4 start-up sequence

IEEE 802.15.4 maintains a process for startup, network configuration, and joining of existing networks. The process is as follows:

1. Device initializes its stack (PHY and MAC layers).
2. PAN coordinator is created. Each network has only one PAN coordinator. The PAN coordinator must be assigned at this phase before proceeding.

3. The PAN coordinator will listen to other networks it has access to and derives a PAN ID that is unique to the PAN it will administer. It can do this over multiple frequency channels.
4. The PAN coordinator will choose a specific radio frequency to use for the network. It will do this using an energy detection scan where it scans the frequencies the PHY can support and listens to find a quiescent channel.
5. The network will be started by configuring the PAN coordinator and then starting the device in coordinator mode. At this point, the PAN coordinator can accept requests.
6. Nodes can join the network by finding the PAN coordinator using an active channel scan where it broadcasts a beacon request across all its frequency channels. When the PAN coordinator detects the beacon, it will respond back to the requesting device. Alternatively, in a beacon-based network (detailed earlier), the PAN coordinator will routinely send out a beacon and the device can perform a passive channel scan and listen for the beacon. The device will then send an association request.
7. The PAN coordinator will determine if the device should or can join the network. This could be based on access control rules, or even if the PAN coordinator has enough resources to manage another device. If accepted, the PAN coordinator will assign a 16-bit short address to the device.

IEEE 802.15.4 security

The IEEE 802.15.4 standard includes security provisions in the form of encryption and authentication. The architect has flexibility in the security of the network based on cost, performance, security, and power. The different security suites are listed in the following table.

AES-based encryption uses a block cipher with a counter mode. The AES-CBC-MAC provides authentication-only protection, and the AES-CCM mode provides the full suite of encryption and authentication. The 802.15.4 radios provide an **access control list (ACL)** to control which security suite and keys to use. Devices can store up to 255 ACL entries.

The MAC layer also computes "freshness checks" between successive repetitions to ensure old frames or old data is no longer considered valid and will stop those frames from proceeding up the stack.

Each 802.15.4 transceiver must manage its own ACL and populate it with a list of "trusted neighbors" along with the security policies. The ACL includes the address of the node cleared to communicate with, the particular security suite to use (AES-CTR, AES-CCM-xx, AES-CBC-MAC-xx), the key for the AES algorithm, and the *last initial vector (IV)* and *replay counter*. The table below lists the various 802.15.4 security modes and features.

Type	Description	Access control	Confidentiality	Frame integrity	Sequential freshness
None	No security				
AES-CTR	Encryption only, CTR	X	X		X
AES-CBC-MAC-128	128-bit MAC	X		X	
AES-CBC-MAC-64	64-bit MAC	X		X	
AES-CBC-MAC-32	32-bit MAC	X		X	
AES-CCM-128	Encryption and 128-bit MAC	X	X	X	X
AES-CCM-64	Encryption and 64-bit MAC	X	X	X	X
AES-CCM-32	Encryption and 32-bit MAC	X	X	X	X

Symmetric cryptography relies on both endpoints using the same key. Keys can be managed at a network level using a shared network key. This is a simple approach where all nodes possess the same key but are at risk of insider attacks. A pairwise keying scheme could be used where unique keys are shared between each pair of nodes. This mode adds overhead, especially for networks where there is a high fan out from nodes to neighbors. Group keying is another option. In this mode, a single key is shared among a set of nodes and is used for any two nodes of the group. Groups are based on device similarities, geographies, and so on. Finally, a hybrid approach is possible, combining any of the three schemes mentioned.

Zigbee

Zigbee is a WPAN protocol based on the IEEE 802.15.4 foundation targeted for commercial and residential IoT networking that is constrained by cost, power, and space. This section details the Zigbee protocol from a hardware and software point of view. Zigbee got its name from the concept of a bee flying. As a bee flies back and forth between flowers gathering pollen, it resembles a packet flowing through a mesh network - device to device.

Zigbee history

The concept of the low-power wireless mesh networking became standard in the 1990s and the Zigbee Alliance was formed to address this charter in 2002. The Zigbee protocol was conceived after the ratification of IEEE 802.15.4 in 2004. That became the IEEE 802.15.4.-2003 standard on December 14, 2004. Specification 1.0, also known as the Zigbee 2004 Specification, was brought public on June 13, 2005. The history can be profiled as:

- 2005 - Zigbee 2004 released
- 2006 - Zigbee 2006 released
- 2007 - Zigbee 2007 released, also known as Zigbee Pro (introduced cluster libraries, some backward compatibility constraints with Zigbee 2004 and 2006)

The alliance's relation to the IEEE 802.15.4 working group is similar to the IEEE 802.11 working group and the Wi-Fi Alliance. The Zigbee Alliance maintains and publishes standards for the protocol, organizes working groups, and manages the list of application profiles. The IEEE 802.15.4 defines the PHY and MAC layers, but nothing above. Additionally, 802.15.4 doesn't specify anything about multi-hop communications or application space. That is where Zigbee (and other standards built on 802.15.4) comes into play.

Zigbee is proprietary and closed standard. It requires a licensing fee and agreement provided by the Zigbee Alliance. Licensing grants the bearer Zigbee compliance and logo certification. It guarantees interoperability with other Zigbee devices.

Zigbee overview

Zigbee is based on 802.15.4 but layers on network services similar to TCP/IP. It can form networks, discover devices, provide security, and manage the network. It does not provide data transport services or an application execution environment. Because it is essentially a mesh network, it is self-healing and ad hoc in form. Additionally, Zigbee prides itself on simplicity and claims a 50% reduction in software support by using a lightweight protocol stack.

There are three principal components in a Zigbee network.

- **Zigbee controller (ZC):** Highly capable device on a Zigbee network that is used to form and initiate network functions. Each Zigbee network will have a single ZC that fulfills the role of an 802.15.4 2003 PAN coordinator (FFD). After the network is formed, the ZC can behave as a ZR (Zigbee router). It can assign logical network addresses and permit nodes to join or leave the mesh.
- **Zigbee router (ZR):** This component is optional but handles some of a load of mesh network hopping and routing coordination. It too can fulfill the role of an FFD and has an association with the ZC. A ZR participates in multi-hop routing of messages and can assign logical network addresses and permit nodes to join or leave the mesh.
- **Zigbee end device (ZED):** This is usually a simple endpoint device such as a light switch or thermostat. It contains enough functionality to communicate with the coordinator. It has no routing logic; therefore, any messages arriving at a ZED that are not targeted to that end device are simply relayed. It also can not perform associations (detailed later).

Zigbee targets three different types of data traffic. Periodic data is delivered or transmitted at a rate defined by the applications (for example, sensors periodically transmitting).

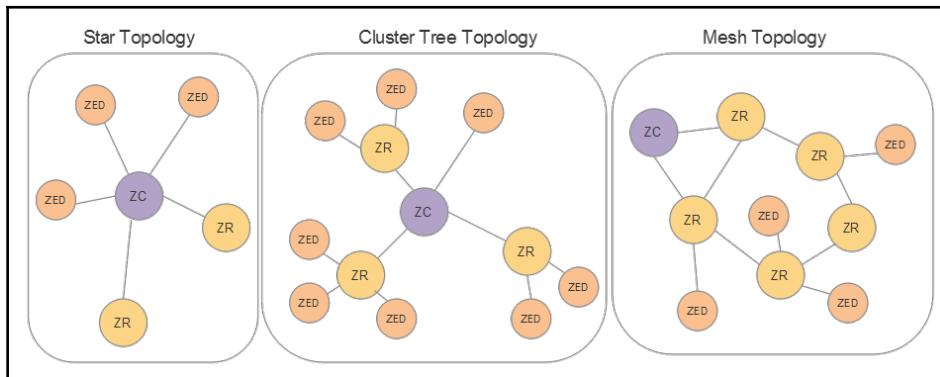
Intermittent data occurs when an application or external stimulus occurs at a random rate. A good example of intermittent data suitable for Zigbee is a light switch. The final traffic type Zigbee serves is repetitive low latency data. Zigbee allocates time slots for transmission and can have very low latency, which is suitable for a computer mouse or keyboard.

Zigbee supports three basic topologies:

- **Star network:** A single ZC with one or more ZEDs. Only extends two hops and is therefore limited in node distance. It also requires a reliable link with a single point of failure at the ZC.
- **Cluster tree:** A multi-hop network that employs beaconing and extends the network coverage and range over a star network. ZC and ZR nodes can have children, but ZEDs remain true endpoints. Child nodes only communicate with their parent (like a small star network). Parents can communicate downstream to its children or upstream to its parent. The problem still exists with a single point of failure at the center.
- **Mesh network:** Dynamic path formation and morphing. Routing can occur from any source device to any destination device. Uses tree and table-driven routing algorithms. ZC and ZR radios must be powered at all times to perform routing duties, consuming battery life. Additionally, calculating the latency in a mesh network can be difficult if not non-deterministic. Some rules are relaxed; however, routers within a certain range of each other can communicate directly with one another. The main advantage is the network can grow beyond the line of sight and has multiple redundant paths.



Zigbee, in theory, can deploy up to 65,536 **Zigbee End Devices (ZED)**.



Three forms of Zigbee network topologies from the simplest star network, to the cluster tree, to a true mesh.

Zigbee PHY and MAC (and difference from IEEE 802.15.4)

Zigbee, like Bluetooth, operates primarily in the 2.4 GHz ISM band. Unlike Bluetooth, it also operates at 868 MHz in Europe and 915 MHz in the USA and Australia. Because of the lower frequency, it has better propensity to penetrate walls and obstacles over traditional 2.4 GHz signals. Zigbee does not use all of the IEEE802.15.4 PHY and MAC specifications. Zigbee does make use of the CSMA/CA collision avoidance scheme. It also uses the MAC level mechanism to prevent nodes from talking over each other.



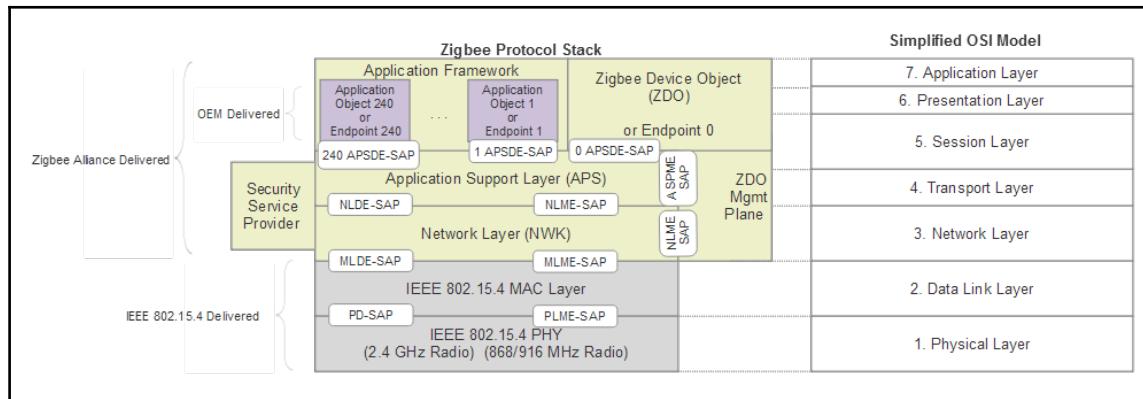
Zigbee does not use the IEEE802.15.4 beaconing modes. Additionally, the guaranteed time slots (GTS) of SuperFrames are also not used in Zigbee.

The security specification of 802.15.4 is slightly modified. The CCM modes that provide authentication and encryption require a different security for every layer. Zigbee is targeted for severely resource-constrained and deeply-embedded systems and does not provide the level of security as defined in 802.15.4.

Zigbee is based on the IEEE 802.15.4-2003 specification before enhancements with two new PHYs and radios were standardized in the IEEE 802.15.4-2006 specification. This implies that the data rates are slightly lower than they could be in the 868 MHz and 900 MHz bands.

Zigbee protocol stack

The Zigbee protocol stack includes a **network layer (NWK)** and an **application layer (APS)**. Additional components include a security service provider, a ZDO management plane, and a **Zigbee device object (ZDO)**. The structure of the stack illustrates genuine simplicity over the more complex but feature filled Bluetooth stack.



Zigbee protocol stack and the corresponding simplified OSI model as a frame of reference.

The NWK is used for all three principal Zigbee components (ZR, ZC, ZED). This layer performs device management and route discovery. Additionally, since it administers a true dynamic mesh, it is also responsible for route maintenance and healing. As the most basic function, the NWK is responsible for transferring network packets and routing of messages. During the process of joining nodes to a Zigbee mesh, the NWK provides the logical network address for the ZC and secures a connection.

The APS provides an interface between the network layer and application layer. It manages the binding table database, which is used to find the correct device depending on the service that is needed versus a service that is offered. Applications are modeled by what are termed *application objects*. Application objects communicate with each other through a map of object attributes called *clusters*. Communication between objects is created in a compressed XML file to allow for ubiquity. All devices must support a basic set of methods; however, a total of 240 endpoints can exist per Zigbee device.

The APS interfaces the Zigbee device to a user. The majority of components in the Zigbee protocol reside here, including the *Zigbee device object* (ZDO). Endpoint 0 is called the ZDO and is a critical component that is responsible for overall device management. This includes managing keys, policies, and roles of devices. It can also discover new (one-hop-away) devices on the network and the services those devices offer. The ZDO initiates and responds to all binding requests for the device. It also establishes a secure relationship between network devices by managing the security policy and keys for the device.

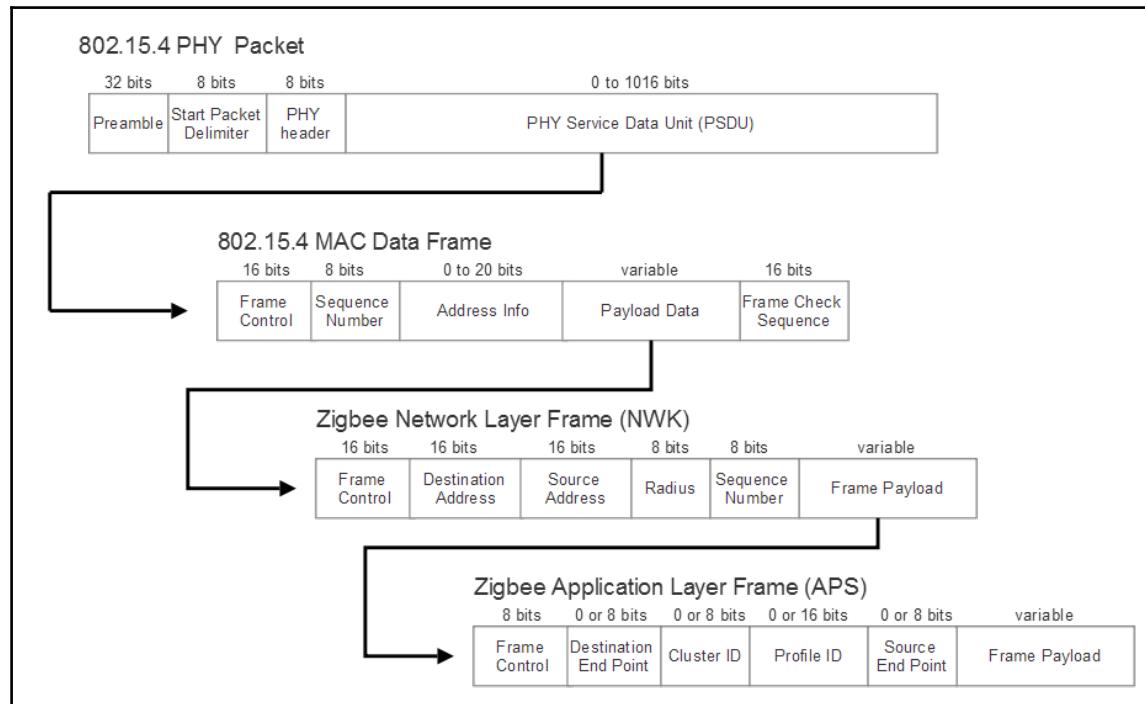
The bindings are connections between two endpoints with each binding supporting a particular application profile. Therefore, when we combine the source and a destination endpoint, the cluster ID, and profile ID, we can create a unique message between two endpoints and two devices. Bindings can be one-to-one, one-to-many, or many-to-one. An example of this binding in action would be multiple light switches connected to a set of light bulbs. The switch application endpoints will associate with light endpoints. The ZDO provides the binding management by associating the switch endpoints to light endpoints using application objects. Clusters can be created, allowing one switch to turn on all the lights while another switch is only able to control a single light.

An application profile delivered by the OEM will describe a collection of devices for specific functions (for example, light switches and smoke alarms). Devices within an application profile can communicate with each other through clusters. Each cluster will have a unique cluster ID to identify itself within the network.

Zigbee addressing and packet structure

The Zigbee protocol, as shown, resides on top of the 802.15.4 PHY and MAC layers and reuses its packet structures. The network diverges at the network and application layers.

The following figure illustrates breaking down one of the 802.15.4 PHY and MAC packets into the corresponding network layer frame packet (NWK) as well as the application layer frame (APS):



Zigbee Network (NWK) and Application Layer (APS) frames residing over 802.15.4 PHY and MAC packets.

Zigbee uses two unique addresses per node:

- **Long address (64 bit):** Assigned by the manufacturer of the device and is immutable. Uniquely identifies the Zigbee device from all other Zigbee devices. This is the same as the 802.15.4 64-bit address. The top 24 bits refer to the organizational unique identifier (OUI) and the bottom 40 bits are managed by the OEM. The addresses are managed in blocks and can be purchased through IEEE.
- **Short address (16 bit):** This too is the same as the PAN ID of the 802.15.4 specification and is also optional.

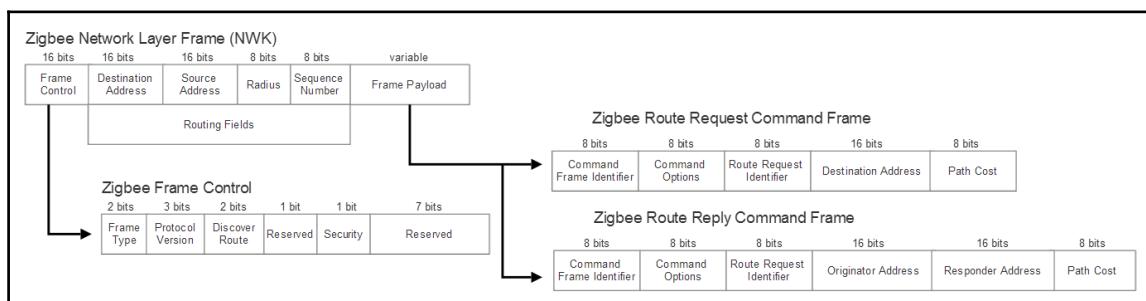
Zigbee mesh routing

Table routing uses Ad Hoc On-Demand Distance Vector Routing (AODV) and the Cluster Tree Algorithm. AODV is a pure on-demand routing system. In this model, nodes don't have to discover one another until there is some association between the two (for example, two nodes need to communicate). AODV also doesn't require nodes that aren't in the routing path to maintain routing information. If a source node needs to communicate to a destination and a path doesn't exist, then a path discovery process starts. AODV provides both unicast and multicast support. It is a reactive protocol; that is, it only provides a route to a destination on demand and not proactively. The entire network is silent until a connection is needed.

The cluster tree algorithm forms a self-organizing network capable of self-repair and redundancy. Nodes in the mesh select a cluster head and create clusters around a head node. These self-forming clusters then connect to each other through a *designated device*.

Zigbee has the ability to route packets in multiple manners:

- **Broadcasting:** Transmit packet to all other nodes in the fabric.
- **Mesh routing (table routing):** If a routing table for the destination exists, the route will follow the table rules accordingly. Very efficient. Zigbee will allow a mesh and table to route up to 30 hops away.
- **Tree routing:** Unicast messaging from one node to another. Tree routing is purely optional and can be disallowed from the entire network. It provides better memory efficiency than mesh routing since a large routing table doesn't exist. Tree routing does not have the same connection redundancy as a mesh, however. Zigbee supports tree routing hops up to 10 nodes away.
- **Source routing:** Used primarily when a data concentrator is present. This is how Z-Wave provides mesh routing.



Zigbee Routing Packet to issue a Route Request Command Frame and subsequent reply with a Route Reply Command Frame.

Route discovery or path discovery is the process of discovering a new route or repairing a broken route. A device will issue a route request command frame to the entire network. If and when the destination receives the command frame, it will respond with at least one route reply command frame. All the potential routes returned are inspected and evaluated to find the optimal route.



Link costs that are reported during path discovery can be constant or based on the likelihood of reception.

Zigbee association

As previously mentioned, **Zigbee end devices (ZED)** do not participate in routing. End devices communicate with the parent, who is also a router. When a **Zigbee coordinator (ZC)** allows for a new device to join a network, it enters a process known as an *association*. If a device loses contact with its parent, the device can at any time rejoin through the process known as *orphaning*.

To formally join a Zigbee network, a *beacon request* is broadcast by a device to ask for subsequent beacons from devices on the mesh that are authorized to allow new nodes to join. At first, only the PAN coordinator is authorized to provide such a request; after the network has grown, other devices may participate.

Zigbee security

Zigbee builds off the security provisions of IEEE 802.15.4. Zigbee provides three security mechanisms: **access control lists (ACL)**, 128-bit AES encryption, and message freshness timers.

The Zigbee security model is distributed across several layers:

- Application layer provides key creation and transport services to ZDO
- Network layer manages routing, and outgoing frames will use the link key defined by the routing if available; otherwise, a network key is used
- MAC layer security is managed through an API and controlled by the upper layers

There are multiple keys that are managed by a Zigbee network:

- **Master key:** The master key may be pre-installed by a manufacturer or entered by a manual process with the user. It forms the basis of security of the Zigbee device. Master keys are always installed first and transmitted from the trust center.
- **Network key:** This key will provide protection at a network level for outside attackers.
- **Link key:** This forms a secure binding between two devices. If two devices have a choice between using installed link keys or network keys, they will always default to link keys to provide more protection.



Link keys are resource heavy in the sense of key storage on constrained devices. Network keys can be used to alleviate some of the storage cost at the risk of decreased security.

Key management is critical for security. The distribution of keys is controlled by establishing a trust center (one node to act as the distributor of keys to all other nodes in the fabric). The ZC is assumed to be the trust center. One may implement a Zigbee network with a dedicated trust center outside of the ZC. The trust center performs the following services:

- Trust management: Authenticates devices joining the network
- Network management: Maintains and distributes keys
- Configuration management: Enables device to device security

In addition, the trust center can be placed in a residential mode (it will not establish keys with network devices) or it can be in commercial mode (establishes keys with every device on the network).

Zigbee uses 128-bit keys as part of its specification within the MAC and NWK layers. The MAC layer provides three modes of encryption, AES-CTR, AES-CBC-128, and AES-CCM-128 (all defined in the IEEE 802.15.4 section). The NWK layer, however, only supports AES-CCM-128 but is tweaked slightly to provide encryption-only and integrity-only protection.

Message integrity ensures messages have not been modified in transit. This type of security tool is used for man-in-the-middle attacks. Referring back to the Zigbee packet structure, the message integrity code and auxilliary header provide fields to add extra checks for each application message sent.

Authentication is provided through a common network key and individual keys between pairs of devices.

Message freshness timers are used to find messages that have timed-out. These messages are rejected and removed from the network as a tool to control replay attacks. This applies to incoming and outgoing messages. Any time a new key is created, the freshness timers are reset.

Z-Wave

Z-Wave is a WPAN protocol used for consumer and home automation primarily, and has about 2100 products using the technology. It has found its way into commercial and building segments in the areas of lighting and HVAC control. In terms of market share, Z-Wave is not at the market cap like Bluetooth or Zigbee. Z-Wave is another mesh technology in the 900 MHz band. Its first manifestation was in 2001 at Zensys, a Danish company developing light control systems. Zensys formed an alliance with Leviton Manufacturing, Danfoss, and Ingersoll-Rand in 2005, formally known as the Z-Wave Alliance. The alliance acquired Sigma Designs in 2008 and Sigma is now the sole provider of Z-Wave hardware modules.

Z-Wave Alliance member companies now include SmartThings, Honeywell, Belkin, Bosch, Carrier, ADT, and LG.

Z-Wave is a closed protocol for the most part with limited hardware module manufacturers. The specification is starting to open up more in the public domain but a substantial amount of material is not disclosed.

Z-Wave overview

Z-Wave's design center is home and consumer lighting/automation. It is intended to use very low bandwidth for communication with sensors and switches. The design is based on the ITU-T G.9959 Standard at the PHY and MAC level. The ITU-T G.9959 is the International Telecommunications Union specification for short-range narrow-band radio-communication transceivers in the sub-1 GHz band.

There are several bands in the sub-1 GHz range that are used for Z-Wave depending on the country of origin. In the US, the 908.40 MHz center frequency is the standard. There are three data rates that Z-Wave is capable of with varying frequency spread for each:

- **100 Kbps:** 916.0 MHz with a 400 KHz spread
- **40 Kbps:** 916.0 MHz with a 300 KHz spread
- **9.6 Kbps:** 908.4 MHz with a 300 KHz spread

Each band operates on a single channel.

Modulation performed at the PHY level uses frequency shift keying for data rates at 9.6 Kbps and 40 Kbps. At the fast 100 Kbps rate, Gaussian frequency shift keying is used. The output power will be roughly 1mW at 0 dB.

Channel contention is managed using CSMA/CA, as described in other protocols previously covered. This is managed in the MAC layer of the stack. Nodes start in receive mode and wait a period of time before transmitting data if there is data being broadcast.

From a role and responsibility point of view, the Z-Wave network is composed of different nodes with specific functions:

- **Controller device:** This top-level device provides the routing table for the mesh network and is the host/master of the mesh under it. There are two fundamental types of controllers:
 - **Primary controller:** The primary controller is the master, and only a single master can exist in a network. It has the ability to maintain the network topology and hierarchy. It can also include or exclude nodes from the topology. It also has the duty of allocating node IDs.
 - **Secondary controller:** These nodes assist a primary controller with routing.
- **Slave device/node:** These devices perform actions based on commands they receive. These devices cannot communicate with neighbor slave nodes unless instructed to do so via a command. Slaves can store routing information but do not compute or update routing tables. Typically, they will act as a repeater in a mesh.

Controllers can also be defined as portable and static. A portable controller is designed to move like a remote control. Once it has changed position, it will recalculate the fastest routes in the network. A static controller is intended to be fixed, such as a gateway plugged into a wall outlet. The static controller can always be "on" and receiving slave status messages.

Controllers can also have different attributes within the network:

- **Status update controller (SUC):** The static controller also has the advantage of taking the role of status update controller. In this case, it will receive notifications from a primary controller regarding topology changes. It can also assist in the routing of slaves.
- **SUC ID server (SIS):** An SUC can also assist in including and excluding slaves for the primary.
- **Bridge controller:** This essentially is a static controller that has the ability to act as a gateway between the Z-Wave mesh and other network systems (for example, WAN or Wi-Fi). The bridge is allowed to control up to 128 virtual slave nodes.
- **Installer controller:** This is a portable controller that can assist in network management and quality-of-service analysis.

Slaves also support different attributes:

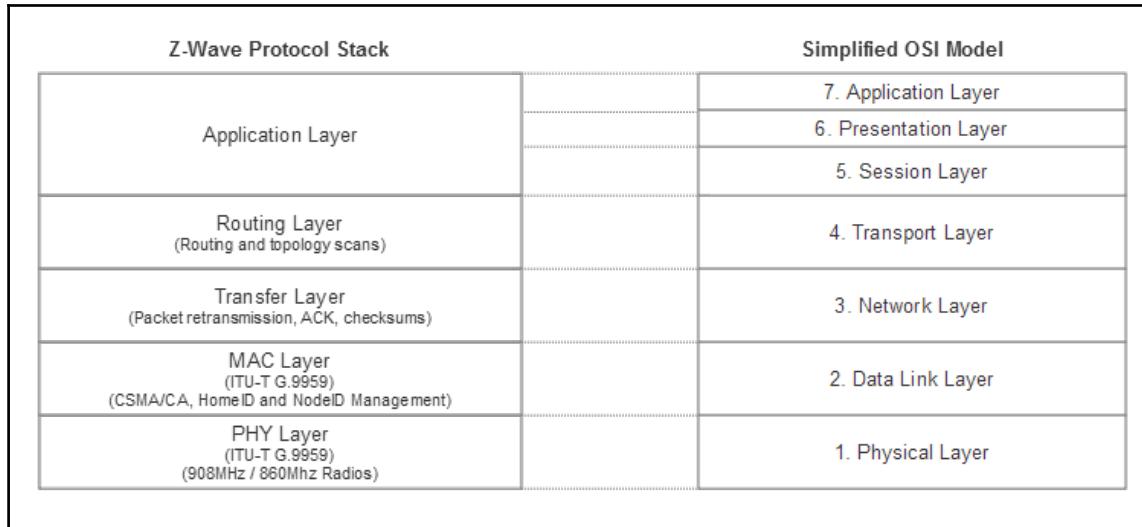
- **Routing slave:** Fundamentally a slave node but with the ability to send unsolicited messages to other nodes in the mesh. Typically, slaves are not allowed to send a message to another node without the command of a primary controller. The node stores a set of static routes that it uses when a message is sent.
- **Enhanced slave:** These have the same abilities as a routing slave with the addition of a real-time clock and persistent storage for application data. An example might be a gas meter.



Slave nodes/devices may be battery based, such as a motion sensor in a home. For a slave to be a repeater, it would always have to be functional and listening for messages on the mesh. Because of this, battery-based devices are never used as repeaters.

Z-Wave protocol stack

Because Z-Wave is a very low bandwidth protocol that is intended to have a sparse network topology, the protocol stack attempts to communicate in as few bytes per message as possible. The stack consists of five layers, as shown in the following figure:



Z-Wave Protocol Stack and OSI model comparison. Z-Wave uses a five-layer stack with the bottom two layers (PHY and MAC) defined by the ITU-T G.9959 Specification.

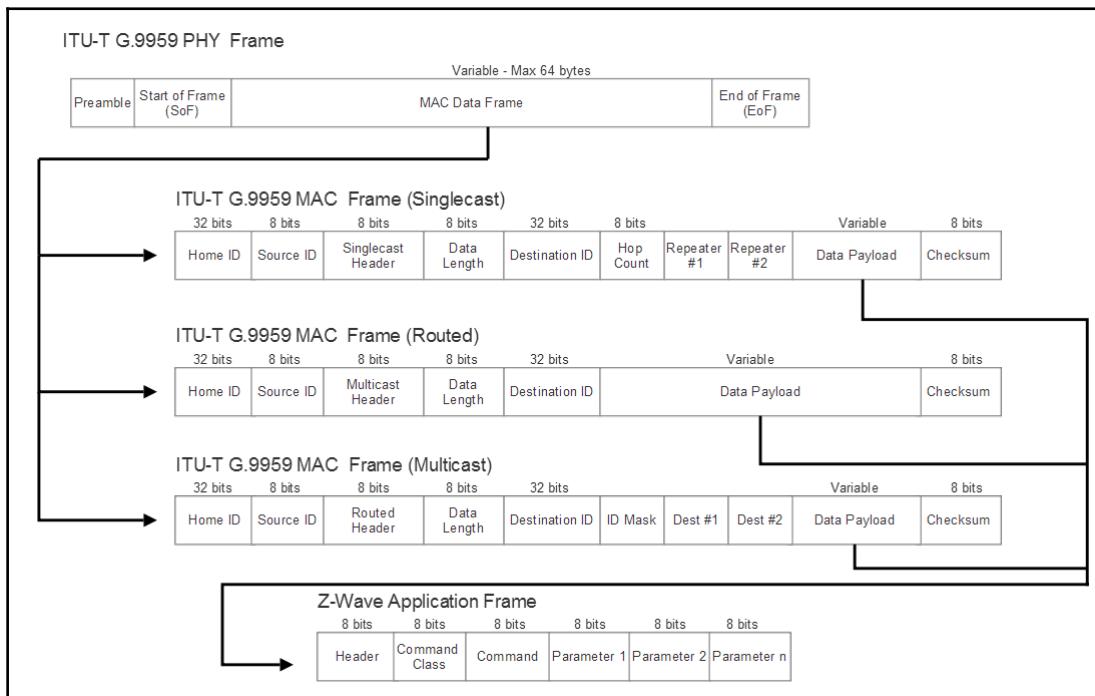
The layers can be described as follows:

- **PHY layer:** Defined by the ITU-T G.9959 Specification. This layer manages the signal modulation, channel assignment, and preamble binding at the transmitter and preamble synchronization at the receiver.
- **MAC layer:** This layer manages the HomeID and NodeID fields, as defined in the previous section. The MAC layer also uses a collision avoidance algorithm and backoff strategy to alleviate congestion and contention on the channel.
- **Transfer layer:** Manages the communication of Z-Wave frames. This layer is also responsible for the retransmission of frames as needed. Additional tasks include acknowledgment of transmissions and checksum binding.
- **Routing layer:** This provides routing services. Additionally, the network layer will perform a topology scan and update of the routing tables.
- **Application layer:** Provides the user interface to applications and data.

Z-Wave addressing

Z-Wave has a fairly simple addressing mechanism compared to Bluetooth and Zigbee protocols. The addressing scheme is kept simple because all attempts are made to minimize traffic and conserve power. There are two fundamental addressing identifiers that need definition before proceeding:

- **Home ID:** This is a 32-bit unique identifier that is preprogrammed in controller devices to assist with identifying Z-Wave networks from each other. During network start, all Z-Wave slaves have a home ID of zero and the controller will systematically populate the slave nodes with the correct home ID.
- **Node ID:** This is an 8-bit value that is assigned to each slave by the controller and provides addressing of slaves in the Z-Wave network.



Z-Wave packet structure from PHY to the Application layer. Three packet types are defined as well: singlecast, routed, and multicast).

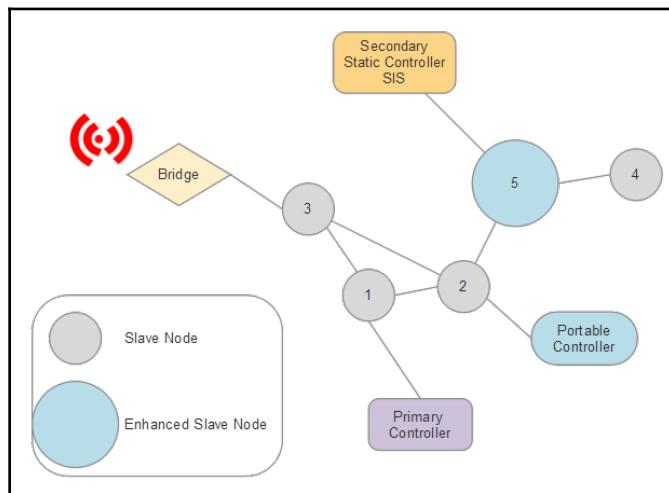
The transport layer provides several frame types to assist with retransmission, acknowledgment, power control, and authentication. The four types of network frames include:

- **Singlecast frame:** This is a packet sent to a single Z-Wave node. This type of packet must be followed by an acknowledgment. If the ACK doesn't occur, a retransmission sequence transpires.
- **ACK frame:** This is the acknowledgment response to a singlecast frame.
- **Multicast frame:** This message is transmitted to more than one node (up to 232). No acknowledgment is used for this type of message.
- **Broadcast frame:** Similar to a multicast message, this frame is transmitted to all nodes in the network. Again, no ACK is used.

For a new Z-Wave device to be used on the mesh, it must undergo a pairing and adding process. The process is usually started by a mechanical or user instantiated keypress on the device. As mentioned, the pairing process involves the primary controller assigning a home ID to the new node. At this point, the node is said to be included.

Z-Wave topology and routing

The topology of the Z-Wave mesh is shown in the following figure using some of the device types and attributes associated with slaves and controllers. A single primary controller manages the network and establishes the routing behaviors:



Z-Wave topology including the single Primary Controller and four slaves and one enhanced slave. The bridge controller acts as a gateway to a WiFi network. A Portable Controller and Secondary Controller also sits on the mesh for assistance to the Primary Controller.

The routing layer of the Z-Wave stack manages the frame transport from one node to another. The routing layer will set up the correct repeater list if one is needed, scan the network for topology changes, and maintain the routing table. The table is fairly simple and simply specifies which neighbor is connected to a given node. It only looks forward one immediate hop. The table is built by the primary controller by asking each node in the mesh what devices are reachable from its location.

The use of source routing to navigate the mesh implies that as the message makes its way through the fabric, for every hop that receives the frame, it will forward the packet to the next node in the chain. As an example, in the following routing table, the shortest path from "Bridge" to "Slave 5" follows this logical path: Bridge | Slave 3 | Slave 2 | Slave 5.

Z-Wave limits the routing hops to a maximum of four.



A routing table of the preceding sample topology is given as:

	Slave 1	Slave 2	Slave 3	Slave 4	Enhanced Slave 5	Primary Controller	Secondary SIS	Bridge	Portable Controller
Slave 1	0	1	1	0	0	1	0	0	0
Slave 2	1	0	1	0	1	0	0	0	1
Slave 3	1	1	0	0	0	0	0	1	0
Slave 4	0	0	0	0	1	0	0	0	0
Enhanced Slave 5	0	1	0	1	0	0	1	0	0
Primary Controller	0	0	0	0	0	0	0	0	0
Secondary SIS	0	0	0	0	1	0	0	0	0
Bridge	1	0	1	0	0	0	0	0	0
Portable Controller	0	1	0	0	0	0	0	0	0

Z-Wave Source-Routing Algorithmic Example



Portable controllers have a challenge in maintaining an optimal routing path; therefore, portable controllers will use alternative techniques to find the best route to a destination node.

Summary

This chapter has covered the first step in delivering IoT data from devices to the internet. The first step in connecting billions of devices is using the correct communication medium to reach sensors, objects, and actuators to cause some action. This is the role of the wireless personal area network. We have explored the basis of the unlicensed spectrum and how as an architect we will measure the performance and behavior of a WPAN. The chapter went deep into the new Bluetooth 5 protocol and compared it to other standards such as the IEEE 802.15.4 basis protocol, Zigbee, and Z-Wave. We explored beaconing, various packet and protocol structures, and mesh architectures. An architect should have an understanding of how these architectures compare and contrast at this point.

The next chapter will explore IP-based PAN and LAN networks such as the ubiquitous 802.11 Wi-Fi network, Thread, and 6LoWPAN, as well as future Wi-Fi communication standards. These networking chapters exploring WPAN, WLAN, and WAN architectures will routinely go back to the fundamentals, such as signal strength measures and range equations taught early in this chapter, and should be used as a reference going forward.

6

IP-Based WPAN and WLAN

WPAN networks have adopted protocols that are typically not TCP/IP, at least from the outset. The protocol stacks for Bluetooth, Zigbee, and Z-Wave have similarities to a true TCP/IP protocol but don't inherently communicate over TCP/IP. There are adaptions of IP on Zigbee (using Zigbee-IP) and IP over Bluetooth (using IPSP to support 6LoWPAN) that do exist. Later in this chapter, we will cover an example of a WPAN using the 802.15.4 protocol with a true IPv6 (thread) compatible layer capable of joining any IPv6 network.

This chapter will also cover the standards around Wi-Fi™ using the IEEE 802.11 protocols. While typically thought of as a wireless LAN (WLAN), 802.11 protocols are pervasive and of use in IoT deployments especially in smart sensors and gateway hubs. This chapter will provide a formal treatment of the 802.11 Wi-Fi catalog of standards including the new IEEE802.11ac high-speed protocol. 802.11 also extends into the world of IoT for the vehicle-to-vehicle and transportation market using the 802.11p protocol which will also be covered. Finally, this chapter will explore the IEEE 802.11ah specification, which is a wireless LAN solution built on the 802.11 protocol but targeted explicitly for power and cost-constrained IoT sensor devices.

The following topics will be covered in the chapter:

- Internet protocol and transmission control protocol
- WPAN with IP – 6LoWPAN
- WPAN with IP – thread
- IEEE 802.11 protocols and WLAN

Internet protocol and transmission control protocol

Supporting an IP layer in a protocol stack does consume resources that could be applied elsewhere. However, there are key benefits in building an IoT system that allows devices to communicate over TCP/IP (transmission control protocol/internet protocol). We will begin by enumerating these benefits; however, it is the role of the architect to balance the cost of these services and features against the impact on a system.

IP role in IoT

From an ecosystem point of view, regardless of the protocol used at a sensor level, the sensor data will ultimately be fed into a public, private, or hybrid cloud for analysis, control, or monitoring. Outside of the WPAN, the world is TCP/IP-based, as we see in WLAN and WAN configurations.

IP is the standard form of global communication for various reasons:

- **Ubiquity:** IP stacks are provided by nearly every operating system and every medium. IP communication protocols are capable of running on various WPAN systems, cellular, copper wire, fiber-optic, PCI Express, and satellite systems. IP specifies the exact format for all data communications and the rules used to communicate, acknowledge, and manage connectivity.
- **Longevity:** TCP was established in 1974, and the IPv4 standard still in use today was designed in 1978. It has withstood the test of time for 40 years. Longevity is paramount for many industrial and field IoT solutions that must support devices and systems for decades. Various other proprietary protocols have been designed by various manufacturers in those 40 years, such as AppleTalk, SNA, DECnet, and Novell IPX, but none have gained the market traction as well as IP.
- **Standards-based:** TCP/IP is governed by the Internet Engineering Task Force (IETF). The IETF maintains a set of open standards focused on the internet protocol.
- **Scalability:** IP has demonstrated scale and adoption. IP networks have demonstrated massive scaling to billions of users and many more devices. IPv6 could provide a unique IP address to every atom comprising Earth and still support 100 more worlds.

- **Reliability:** IP at its heart is a reliable protocol for data transmission. It accomplishes this through a packet delivery system based on a connectionless network. The service is considered unreliable from conception, meaning the data is not guaranteed to be delivered. IP is connectionless because each packet is treated independently from one another. The IP is also referred to as *best-effort delivery* because all attempts will be made to transmit a packet through various routes. The strength of this model allows an architect to replace the delivery mechanism with another—essentially replacing layers one and two of the stack with something else (for example, Wi-Fi with cellular).
- **Manageability:** Various tools exist to manage IP networks and devices on an IP network. Modeling tools, network sniffers, diagnostic tools, and various appliances exist to assist in building, scaling, and maintaining networks.

The transport layer is also worth considering. While IP addresses the need for a well supported and robust network layer, TCP and **Universal Datagram Protocol (UDP)** are needed for the transport layer. The transport layer is responsible for end-to-end communication. The logical communication between different hosts and various network components is governed at this level. TCP is used for connection-oriented transmissions, whereas UDP is used for connectionless transmissions. UDP is naturally much simpler to implement than TCP, but not as resilient. Both services provide segment reordering as packets are not guaranteed to be delivered in order using an IP protocol. TCP also provides the layer of reliability to an unreliable IP network layer through the use of acknowledgment messages and retransmissions of lost messages. Additionally, TCP provides flow control using sliding windows and congestion avoidance algorithms. UDP provides a lightweight, high-speed method to broadcast data to various devices that may or may not be present or reliable.

The following is the standard 7-layer Open Source Interconnection Model stack.

OSI (Open Source Interconnection) Model			
Layers	Purpose / Function	Protocol Used	Fundamental Data Type
7. Application	User Application Layer: browser, ftp, app, etc. (remote file access, resource sharing, LDAP, SNMP)	SMTP FTP	Data
6. Presentation	Syntax Layer: encrypt, compress (optional) (data encrypt/decrypt, codec, translation)	JPEG, ASCII, ROT13	Data
5. Session Layer	Synchronization & Logical Port Routing (session establishment, start & terminate, security, logging, name recognition)	RPC, NFS, NetBIOS	Data
4. Transport Layer	TCP: Host to Host & Flow Control (end to end connections & reliability, message segmentation, acknowledgment, session multiplexing)	TCP / UDP	TCP: Segments UDP: Datagrams
3. Network Layer	Packets: IP Address (path determination, logical addressing, routing, traffic control, frame segmentation, subnet management)	IP, IPX, ICMP	Packets
2. Data Link Layer	Data Frames: MAC address, packet (physical addressing, Media Access Control, LLC, frame error checking, sequencing and reordering)	PPP/SLIP	Frames
1. Physical Layer	Physical Device: Cables, fibers, RF spectrum (data encoding, media attachment, baseband/broadband, signaling, binary transmission)	Coax, fiber, wireless	Bits / Signals

Full seven-layer OSI model. TCP/IP represent layers three and four.

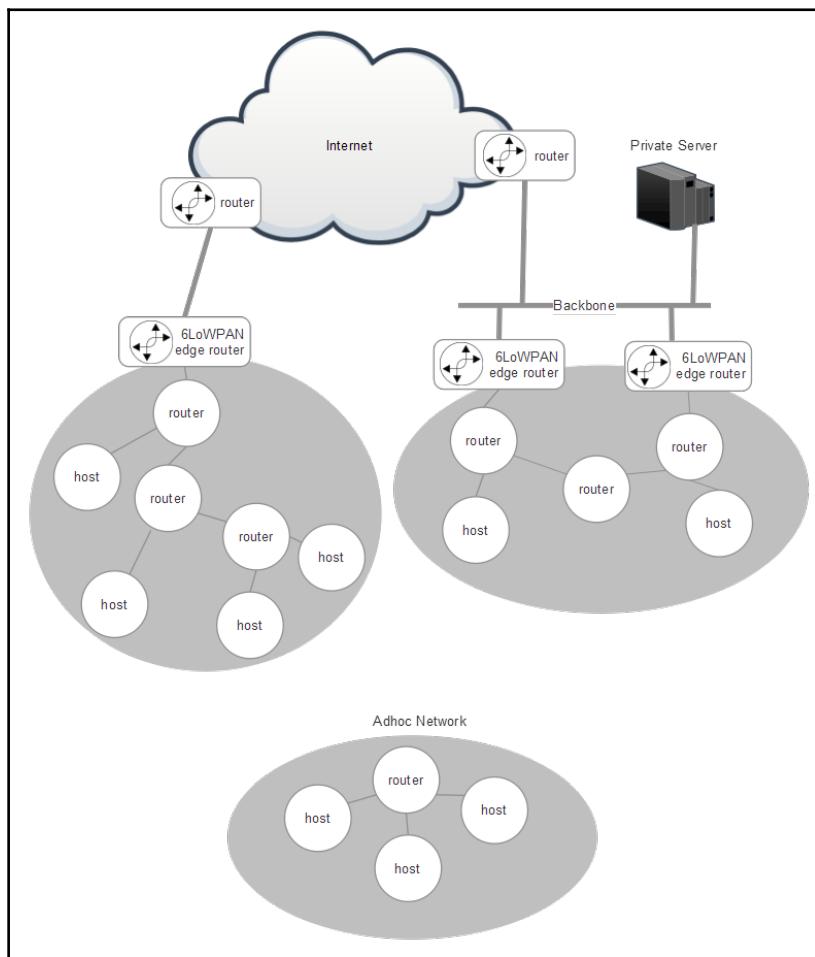
From an IoT perspective, bringing IP close to the source of data bridges two worlds of data management. The **Information Technology (IT)** role manages the infrastructure, security, and provisioning of networks and things on the network. The **Operational Technology (OT)** role manages the health and throughput of the system that functions to produce something. These two roles have traditionally been separated, as things such as sensors, meters, and programmable controllers have not been connected, at least directly. Proprietary standards have governed the OT systems, at least from an industrial IoT perspective.

WPAN with IP – 6LoWPAN

In an effort to bring IP addressability to the smallest and most resource-constrained devices, the concept of 6LoWPAN was formed in 2005. A working group formalized the design in the IETF under the specification RFC 4944 (request for comment) and later updated with RFC 6282 to address header compression and RFC 6775 for neighbor discovery. The consortium is closed, however, the standard is open for anyone to use and implement. 6LoWPAN is an acronym that stands for IPV6 over low power WPANs. The intent is for IP networking over low-power RF communication systems for devices that are power and space constrained and do not need high bandwidth networking services. The protocol can be used with other WPAN communications such as 802.15.4 as well as Bluetooth, sub-1 GHz RF protocols, and **power line controller (PLC)**. The principal advantage of 6LoWPAN is that the simplest of sensors can have IP addressability and act as a network citizen over 3G/4G/LTE/Wi-Fi/Ethernet routers. A secondary effect is that IPV6 provides significant theoretical addressability of 2^{128} or 3.4×10^{38} unique addresses. This would sufficiently cover the estimates of 50 billion internet-connected devices by 2020 and continue to cover those devices well beyond that. Therefore, IPV6 is well-suited for IoT growth.

6LoWPAN topology

6LoWPAN networks are mesh networks residing on the periphery of larger networks. The topologies are flexible, allowing for ad hoc and disjointed networks without any binding to the internet or other systems, or they can be connected to the backbone or the internet using edge routers. 6LoWPAN networks can be conjoined with multiple edge routers; this is called **multi-homing**. Additionally, ad-hoc networks can form without requiring an Internet connectivity of an edge router. These topologies are shown below:



6LoWPAN Topologies.

An edge router (also known as border router) is necessary for a 6LoWPAN architecture as it has four functions:

- Handles the communication to the 6LoWPAN devices and relays data to the internet.
- Performs compression of IPv6 headers by reducing a 40-byte IPv6 header and 8-byte UDP headers for efficiency in a sensor network. A typical 40-byte IPv6 header can compress to two to 20-bytes depending on usage.
- Initiates the 6LoWPAN network.
- Exchanges data between devices on the 6LoWPAN network.

Edge routers form 6LoWPAN mesh networks on larger traditional network perimeters. They can also broker exchanges between IPV6 and IPV4 if necessary. Datagrams are handled in a similar manner as in an IP network, which has some advantages over proprietary protocols. All nodes within a 6LoWPAN network share the same IPv6 prefix that the edge router establishes. Nodes will register with the edge routers as part of the **Network Discovery (ND)** phase.

ND controls how hosts and routers in the local 6LoWPAN mesh will interact with each other. Multi-homing allows for multiple 6LoWPAN edge routers to manage a network; for example, when there is a need for multiple media (4G and Wi-Fi) for failover or fault tolerance.

There are three types of nodes within the 6LoWPAN mesh:

- **Router nodes:** These nodes marshal data from one 6LoWPAN mesh node to another. Routers can also communicate outward to the WAN and internet.
- **Host nodes:** Hosts in the mesh network cannot route data in the mesh and are simply endpoints consuming or producing data. Hosts are allowed to be in sleep states, occasionally waking to produce data or receive data cached by their parent routers.
- **Edge routers:** As stated, these are the gateways and mesh controllers usually at a WAN edge. A 6LoWPAN mesh would be administered under the edge router.

Nodes are free to move and reorganize/reassemble in a mesh. For that matter, a node can move and associate with a different edge router in a multi-home scenario or even move between different 6LoWPAN meshes. These changes to the topology can be caused for various reasons, such as changes in signal strength or physical movement of nodes. When a topology change occurs, the IPv6 address of the associated nodes will also naturally change.

In an ad hoc mesh without an edge router, a 6LoWPAN router node could manage a 6LoWPAN mesh. This would be the case when WAN connectivity to the internet is not necessary. Typically, this is seldom seen as IPv6 addressability for a small ad hoc network isn't necessary. The router node would be configured to support two mandatory functions:

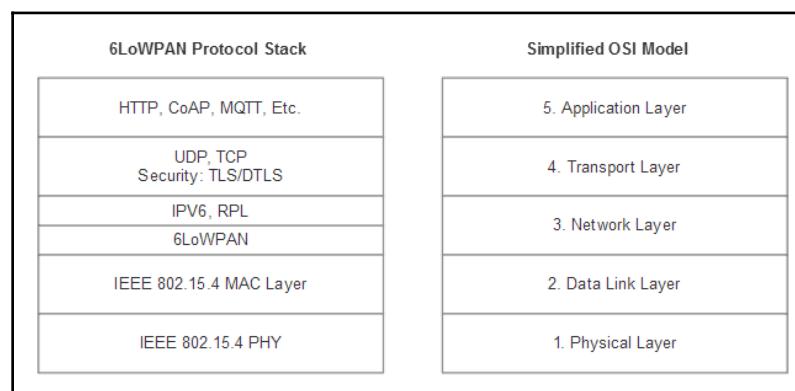


- Unique local unicast address generation
- Performing neighbor discovery ND registration

The ad hoc mesh IPv6 prefix would be a local prefix rather than the larger global WAN IPv6 prefix.

6LoWPAN protocol stack

To enable 6LoWPAN on a form of communication media such as 802.15.4 there is a set of recommended features necessary to support an IP protocol. These features include framing, unicast transmission, and addressing. As shown in the following figure, the physical layer is responsible for receiving and converting data bits over the air. For this example, the link layer we speak to is IEEE 802.15.4. On top of the physical layer is the data link layer, responsible for detecting and correcting errors on the physical link. Detailed information on the physical and data link layer of 802.15.4 are covered earlier in this chapter. Here is the comparison between the 6LoWPAN stack and the OSI model:



6LoWPAN Protocol Stack Comparison to the simplified OSI model. 6LoWPAN resides on top of other protocols like 802.15.4 or Bluetooth to provide the physical and MAC address.

By enabling IP traffic at a sensor level, the relationship between the device and the gateway would use some form of application layer to convert data from the non-IP protocol to an IP-protocol. Bluetooth, Zigbee, and Z-Wave all have some form of conversion from their base protocol to something that can communicate over IP (if the intent is to route the data). The edge routers forward datagrams at a network layer. Because of this, the router does not need to maintain application state. If an application protocol changes, a 6LoWPAN gateway doesn't care. If an application protocol changes for one of the non-IP protocols, the gateway will need to change its application logic as well. 6LoWPAN provides an adaptation layer within layer three (network layer) and on top of layer two (data link layer). This adaptation layer is defined by the IETF.

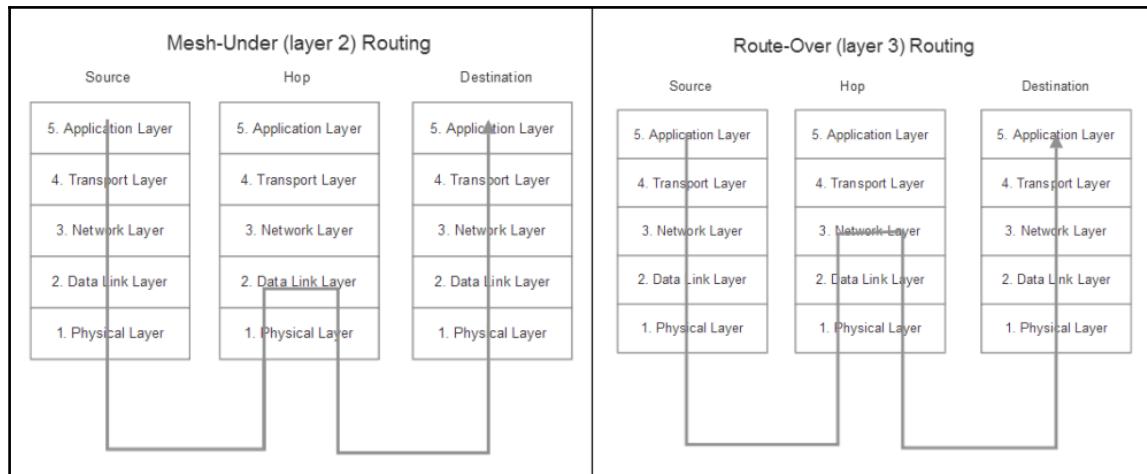
Mesh addressing and routing

Mesh routing operates in the physical and data link layers to allow packets to flow through a dynamic mesh using multiple hops. We previously talked about mesh-under and route-over routing, but we will go somewhat deeper in this section on that form of routing.

6LoWPAN mesh networks utilize two schemes for routing:

- **Mesh-under network:** In a mesh-under topology, routing is transparent and assumes a single IP subnet representing the entirety of the mesh. A message is broadcast in a single domain and is sent to all devices in the mesh. As previously mentioned, this generates considerable traffic. Mesh-under routing will move from hop to hop in the mesh but only forward packets up to layer two (data link layer) of the stack. 802.15.4 handles all the routing for each hop in layer two.
- **Route-over network:** In a route-over topology, networks will incur the charge of forwarding packets up to layer three (network layer) of the stack. Route-over schemes manage routes at an IP level. Each hop represents one IP router.

The following graphic depicts the difference between mesh-under and route-over routing:



The difference between mesh-under and route-over networking. The intermediary hops reveal how far up each stack the packet is delivered before moving to the next node in the mesh.

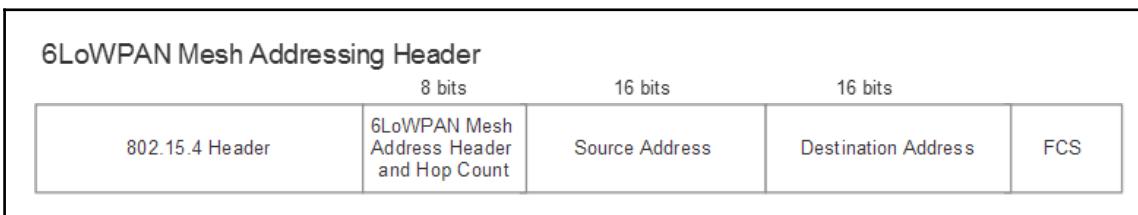
A route-over network implies that every router node is equally capable and can perform the larger set of functions as a normal IP router, such as duplicate address detection. RFC6550 formally defines the route-over protocol RPL (ripple). The advantage of the route-over architecture is the similarity with traditional TCP/IP communication. RPL provides multipoint-to-point communication (where traffic from devices in a mesh communicate to a central server on the internet) and point-to-multipoint communication (central service to the devices in the mesh).

The RPL protocol has two modes for managing the route tables:

- **Storing mode:** All devices configured as routers in a 6LoWPAN mesh maintain routing and neighbor tables.
- **Non-storing mode:** Only a single device, such as the edge router, maintains the routing and neighbor tables. To transmit data from one host to another in a 6LoWPAN mesh, the data is sent up to the router where its route is calculated then transmitted to the receiver.

The routing table, as the name implies, contains the mesh routing paths, while the neighbor table maintains each node's directly connected neighbors. This implies that the edge router will always be referenced to deliver packets in the mesh. This allows the router nodes the freedom of not managing large routing tables to manage, but does add some latency in moving packets since the edge router must be referenced. Storing mode systems will have higher processing and memory requirements to manage routing tables stored on each node, but will have a more efficient path to establishing a route.

Note in the following figure the hop count, source address, and destination address fields. These fields are used during the address resolution and routing phase. The hop count is set to an initial high value and then decremented each time the packet propagates from node to node in the mesh. The intent is when the hop limit reaches zero, the packet is dropped and lost from the mesh. This provides a way to prevent runaway networks in the case a host node removes itself from the mesh and is no longer reachable. The source and destination address are 802.15.4 addresses and may be in short or extended format, as 802.15.4 allows. The header is constructed as:



6LoWPAN Mesh Addressing Header

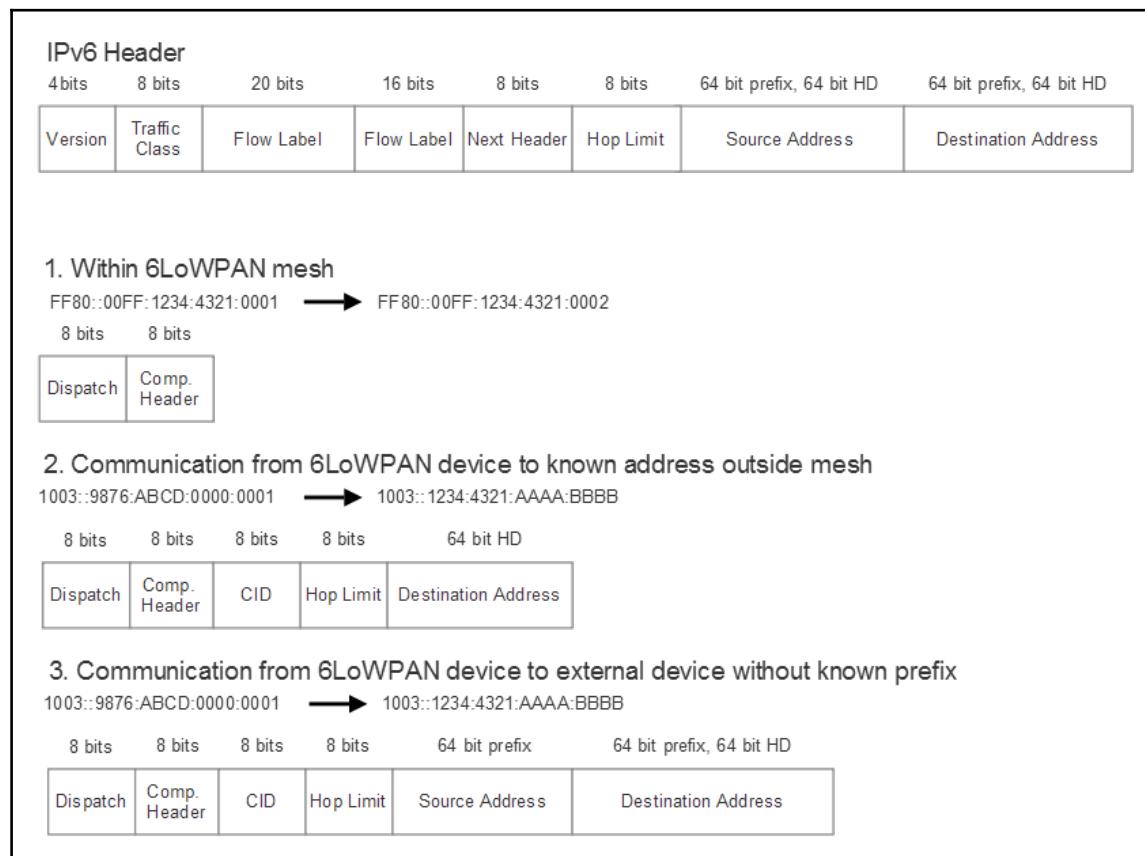
Header compression and fragmentation

While the advantage of having virtually unlimited IP addresses for things is a significant milestone, placing IPv6 on an 802.15.4 link poses some challenges that must be overcome to make 6LoWPAN usable. First is the fact that IPv6 has a **Maximum Transmission Unit (MTU)** size of 1280 bytes, while 802.15.4 has a limit of 127 bytes. The second issue is that IPv6 in general adds significant girth to an already bloated protocol. For example, in IPv6 headers are 40 bytes long.



Note IEEE 802.15.4g does not have a limitation of 127 bytes for a frame length.

Header compression is a means to compress and remove redundancy in the IPv6 standard header for efficiency reasons. Normally, header compression is status-based, meaning in a network with static links and stable connections, it works reasonably well. In a mesh network such as 6LoWPAN, this will not work. Packets hop between nodes and would require compression/decompression on each hop. Additionally, the routes are dynamic and allowed to change and transmissions may not be present for long duration. Therefore, 6LoWPAN adopted stateless and shared-context compression. The type of compression can be affected by whether or not certain specifications are met, such as using RFC4944 over RFC6922, as well as where the source and destination of a packet reside, shown as follows:



Header Compression in 6LoWPAN.

The three cases of header compression for 6LoWPAN based on whether the route is within the local mesh, outside of mesh but to a known address, or outside of mesh to an unknown address. Compared to standard IPv6 with a 40-byte header, 6 LoWPAN can compress between 2 to 20 bytes.

Case one (in the preceding figure) is the best case communication between nodes in a local mesh. No data is sent outward to the WAN with this compressed header format. The second case implies data is sent outward to the WAN to a known address, and the last case is similar but to an address that is known. Even in the third case, which is the worst case, compression still accounts for a 50 percent reduction in traffic. 6LoWPAN also allows for UDP compression, which is beyond the scope of this book.

Fragmentation is a secondary issue, due to the fact that the MTU sizes are incompatible between 802.15.4 (127 bytes) and IPv6 at 1280 bytes. The fragmentation system will divide each IPv6 frame into smaller segments. On the receiving side, the fragments will be reassembled. Fragmentation will differ depending on the type of routing elected during mesh configuration (we will discuss mesh-under and route – over routing later). The types of fragmentation and constraints are given as:

- **Mesh-under routing fragmentation:** Fragments will be reassembled only at the final destination. All fragments need to be accounted for during re-assembly. If any are missing, the entire packet needs retransmission. As a side note, mesh-under systems require all fragments to be transmitted immediately. This will generate a burst of traffic.
- **Route-over routing fragmentation:** Fragments will be reassembled at every hop in the mesh. Each node along the route carries enough resources and information to rebuild all fragments.

The fragmentation header includes a Datagram Size field, which specifies the total size of the unfragmented data. The Datagram Tag field identifies the set of fragments belonging to a payload, and the Datagram Offset indicates where the fragment belongs in a payload sequence. Note, the datagram offset is not used for the first fragment sent as the offset of a new fragment sequence should start at zero.

6LoWPAN Fragmentation Header					
8 bits	8 bits	16 bits			
802.15.4 Header	6LoWPAN Fragmentation Header	Datagram Size	Datagram Tag	Datagram Offset	FCS

6LoWPAN Fragmentation Header

Fragmentation is a resource-heavy task that demands processing and energy ability that can tax a battery-based sensor node. It is advisable to limit the data sizes (at an application level) and use header compression to reduce the power and resource constraints in a large mesh.

Neighbor discovery

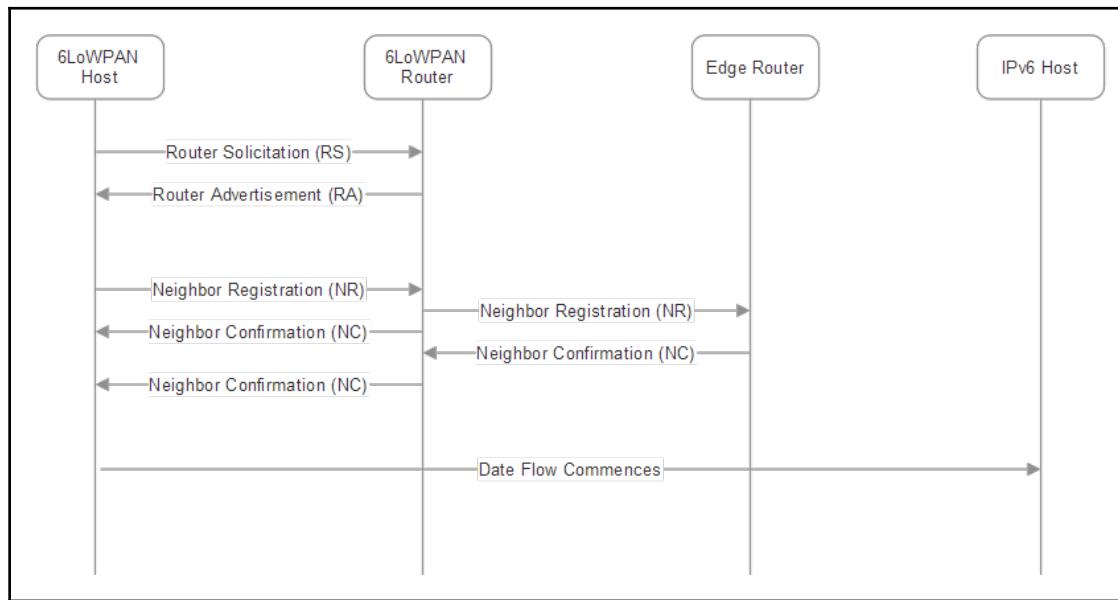
Neighbor discovery (ND) is defined by RFC4861 as a one-hop routing protocol. This is a formal contract between neighbor nodes in the mesh and allows nodes to communicate with each other. ND is the process of discovering new neighbors, as a mesh can grow, shrink, and transform, resulting in new and changing neighbor relations. There are two basic processes and four basic message types in ND:

- **Finding neighbors:** This includes **Neighbor Registration (NR)** and **Neighbor Confirmation (NC)** phases
- **Finding routers:** This includes **Router Solicitation (RS)** and **Router Advertisement (RA)** phases

During ND, there can be conflicts that arise. For example, if a host node disassociates with a router and establishes a link with a different router in the same mesh. ND is required to find duplicate addresses and unreachable neighbors as part of the specification. DHCPv6 can be used in conjunction with neighbor discovery.

After an 802.15.4-capable device has bootstrapped through the physical and data link layers, 6LoWPAN can perform neighbor discovery and grow the mesh. This process will proceed as follows and as shown in the subsequent figure:

1. Finding an appropriate link and subnet for low-power wireless.
2. Minimizing node-initiated control traffic.
3. The host sending out RS message to request the mesh network prefix.
4. The router responding with prefix.
5. The host assigning itself a link-local unicast address (`FE80::IID`).
6. The host transmitting this link-local unicast address in an NR message to the mesh.
7. Performing **Duplicate Address Detection (DAD)** by waiting for an NC for a set time. If the timeout expires, assumes the address is unused.



Simplified neighbor discovery sequence from a 6LoWPAN mesh node through a mesh router to the edge router and then off to the wide area network.

After the host is configured, it can begin communicating over the internet with a unique IPv6 address.

If mesh-under routing is used, the link-local address retrieved in step five can be used to communicate to any other node in the 6LoWPAN mesh. In a route-over scheme, the link-local address can be used to communicate only with nodes that are a single hop away. Anything greater than one hop will require a full routable address.

6LoWPAN security

Since, in a WPAN system, it is easy to sniff and overhear communication, 6LoWPAN provides for security at multiple levels. At the 802.15.4 level two of the protocol, 6LoWPAN relies on AES-128 encryption of data. Additionally, 802.15.4 provides a counter with CBC-MAC mode (CCM) to provide encryption and an integrity check. Most chipsets that provide an 802.15.4 network block also include a hardware encryption engine for performance improvement.

At layer three (the network layer) of the protocol, 6LoWPAN has the option to use IPsec standard security (RFC4301). This includes:

- **Authentication Handler (AH)**: As defined in RFC4302 for integrity protection and authentication
- **Encapsulating Security Payload (ESP)**: In RFC4303, adds encryption to secure confidentiality in packets

ESP is by far the most common layer-three secure packet format. Additionally, a mode of ESP defines reusing AES/CCM used in layer-two hardware for layer-three encryption as well (RFC4309). This makes layer three security suitable for constrained 6LoWPAN nodes.

In addition to link layer security, 6LoWPAN also utilizes **Transport Layer Security (TLS)** for TCP traffic and **Datagram Transport Layer Security (DTLS)** for UDP traffic.

WPAN with IP – Thread

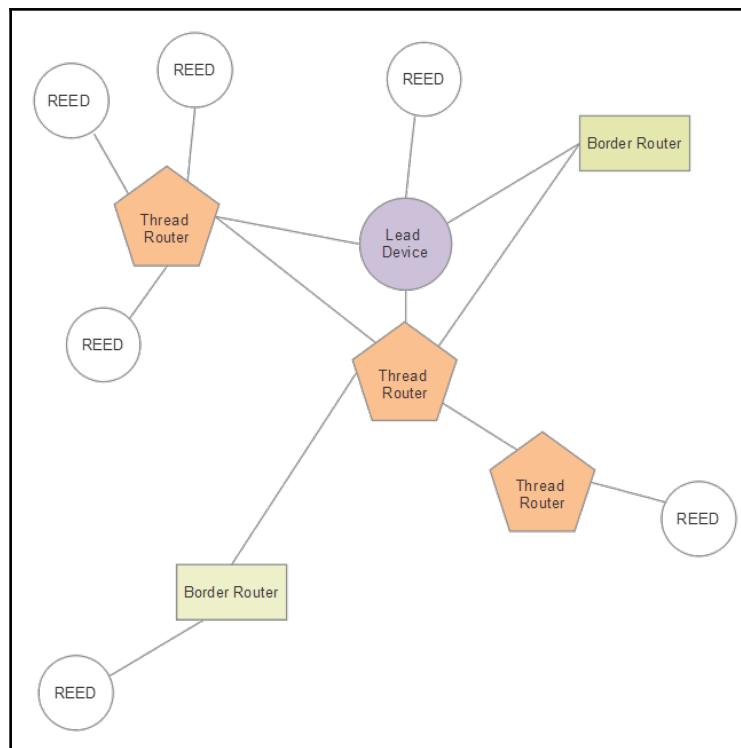
Thread is a relatively new networking protocol for IoT and is based on IPV6 (6LoWPAN). Its principal target is home connectivity and home automation. Thread was launched in July of 2014 with the formation of the Thread Group Alliance, which includes companies such as Alphabet (Google holding company), Qualcomm, Samsung, ARM, Silicon Labs, Yale (locks), and Tyco.

Based on the IEEE 802.15.4 protocol and 6LoWPAN, it has commonality with Zigbee and other 802.15.4 variants, but with a significant difference being Thread is IP addressable. This IP protocol builds on the data and physical layers provided by 802.15.4 and the features such as security and routing from 6LoWPAN. Thread is also mesh-based, making it attractive for home lighting systems with up to 250 devices in a single mesh. The philosophy with Thread is that by enabling IP addressability in the smallest of sensors and home automation systems, one can reduce power because it doesn't need to persist application state since the protocol uses datagrams at the network layer. This also implies that the edge router hosting a Thread mesh network doesn't need to process application layer protocols and can lower its power and processing needs. Finally being IPV6 compliant, it is inherently secure with all communications being encrypted using the **Advanced Encryption Standard (AES)**. Up to 250 nodes can exist on a Thread mesh all with fully encrypted transport and authentication. A software upgrade allows a pre-existing 802.15.4 device to be Thread compatible.

Thread architecture and topology

Based on the IEEE 802.15.4-2006 standard, Thread uses the specification to define the **Medium Access Controller (MAC)** and physical (PHY) layers. It operates at 250 Kbps in the 2.4 GHz band.

From a topology point of view, Thread establishes communications with other devices through a border router (usually a Wi-Fi signal in a household). The rest of the communication is based on 802.15.4 and forms a self-healing mesh. An example of such a topology is shown as follows:



Example Thread network topology containing border routers, Thread routers, lead devices, and eligible IoT devices that can combine in the mesh. Interconnections are variable and self-healing.

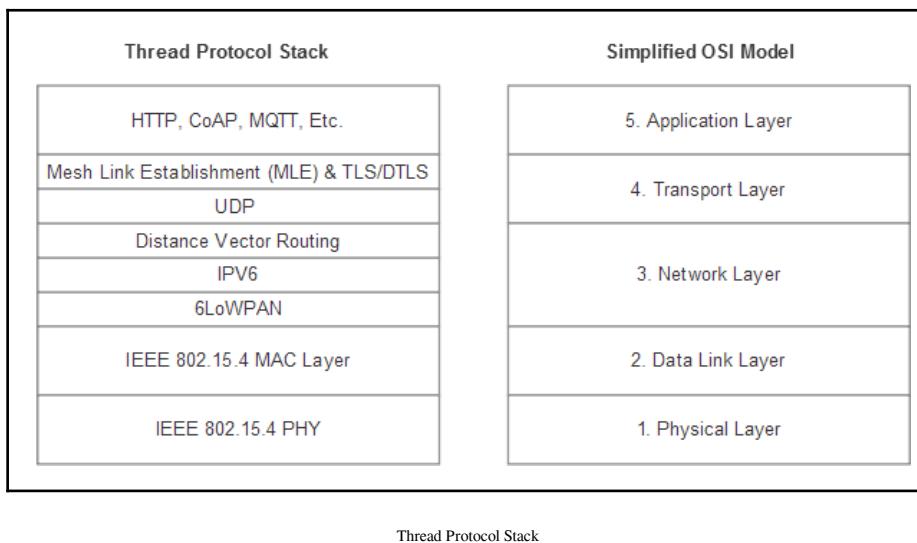
The following are the roles of various devices in a Thread architecture.

- **Border router:** A border router is essentially a gateway. In the home network, this would be a communications crossover from Wi-Fi to Thread and forms the entry point to the internet from a Thread mesh running underneath a border router. Multiple border routers are allowable under the Thread specification.
- **Lead device:** The lead device manages a registry of assigned router IDs. The lead also controls the requests for **Router-eligible End Devices (REED)** to be promoted to routers. A leader can also act as a router and have device-end children. The protocol for assignment of router addresses is the **Constrained Application Protocol (CoAP)**. The state information a lead device manages can also be stored in the other thread routers. This allows for self-healing and failover in case the leader loses connectivity.
- **Thread routers:** Thread routers manage the routing services of the mesh. Thread routers never enter a sleep state but are allowed by the specification to downgrade themselves to become a REED.
- **REEDs:** A host device that is a REED can become routers or a leader. REEDs are not responsible for routing in the mesh unless they are promoted to a router or leader. REEDs also cannot relay messages or join devices to the mesh. REEDs essentially are endpoints or leaf nodes in the network.
- **End devices:** Some endpoints cannot become routers. These types of REEDs have two other categories that they can subscribe to: **full end devices (FEDs)** and **minimal end devices (MEDs)**.
- **Sleepy end devices:** Host devices that have entered a sleep state communicate only with their associated thread router and cannot relay messages.

Thread protocol stack

Thread will make use of the full benefits of 6LoWPAN and enjoy the benefits of header compression, IPv6 addressing, and security. Thread also uses the fragmentation scheme of 6LoWPAN as described in the previous section, but adds two additional stack components:

- Distance vector routing
- Mesh link establishment



Thread Protocol Stack

Thread routing

Thread uses route-over routing as described in the previous section, on 6LoWPAN routing. Up to 32 active routers are allowed in a Thread network. Route traversal is based on next-hop routing. The master route table is maintained by the stack. All routers have an up-to-date copy of the routing for the network.

Mesh link establishment (MLE) is a method to update path traversal costs from one router to another in a network. Additionally, MLE provides a way to identify and configure neighboring nodes in the mesh and secure them. As a mesh network can expand, shrink, and change shape dynamically, MLE provides the mechanism to reconstitute the topology. MLE exchanges the path costs with all other routers in a compressed format. MLE messages will flood the network in a broadcast manner by the **Multicast Protocol for Low Power and Lossy Networks (MPL)**.

Typical 802.15.4 networks use on-demand route discovery. This can be costly (bandwidth due to route discovery flooding the network), and Thread attempts to avoid that scheme. Periodically, a Thread network router will exchange MLE advertisement packets with link cost information to its neighbors essentially forcing all routers to have a current path list. If a route terminates (host leaves Thread network) then the routers attempt to find the next best path to a destination.

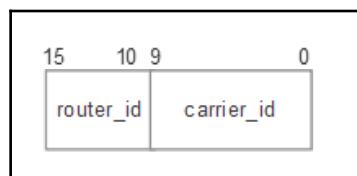
Thread also measures the link quality. Remember that 802.15.4 is a WPANs and signal strength may change dynamically. Quality is measured by the link cost of incoming messages for a neighbor with a value of zero (unknown cost) to three (good quality). The following table summarizes the quality to cost relation. This quality and cost are continually monitored and, as mentioned, periodically broadcast to the network for self-healing:

Link quality	Link cost
0	Unknown
1	6
2	2
3	1

Thread addressing

To find a route to a child node, the path simply inspects the high bits of the child's address to find the parent router address. At this point, the source of transmission knows the cost to reach the child as well as the next-hop information to start the route.

Distance vector routing is used to find paths to routers in the Thread network. The upper 6 bits of the 16-bit address indicate the destination router—this is the prefix. If the lower 10 bits of the destination are set to 0, then the final destination is that router. Alternatively, the destination router will forward the packet on based on the lower 10 bits.



2-Byte Thread Short Address per 802.15.4-2006 specification

If a route extends outside of the Thread network, then a border router will signal to leaders of the particular prefix data, including prefix data, 6LoWPAN context, border routers, and the DHCPv6 server. This information is communicated via MLE packets through the Thread network.

Within the Thread network, all addressing is UDP-based. If a retry is necessary, then the Thread network will rely on:

- **MAC level retries:** Where each device using MAC acknowledgments is an ACK is not received from the next hop
- **Application retries:** Where the application layer will provide its own retry mechanism

Neighbor discovery

Neighbor discovery (ND) in thread decides which 802.15.4 network to join. The process is as follows:

1. Joining device contacts router for commissioning.
2. Joining device scans all channels and issues a beacon request on each channel. Waits for beacon response.
3. If a beacon containing the payload with the network **Service Set Identifier (SSID)** and permit-joining message is seen, the device will now join the Thread network.
4. Once a device is discovered, MLE messages will be broadcast to identify a neighbor router to the device. That router will perform commissioning. There are two commissioning modes:
 - **Configuring:** Uses an out-of-band method to commission a device. Allows for a device to attach to the Thread network as soon as it is introduced to the network.
 - **Establishing:** Creates a commissioning session between the device and a commission application running on a Smartphone, tablet, or web-based.
5. The device that is joining contacts the parent router and attaches to the network via MLE exchange.

The device will exist as either a REED or an end device and is allocated a 16-bit short address by the parent.

IEEE 802.11 protocols and WLAN

One of the first adopters of the ISM bands that the FCC freed for unlicensed use was the IEEE 802.11 technology. The IEEE 802.11 is a suite of protocols with a rich history and different use cases. 802.11 is the specification defining the **Media Access Controller (MAC)** and physical layer (PHY) of a networking stack. The definition and specifications are governed by the IEEE LAN/MAN Standards Committee. Wi-Fi is the definition of WLAN based on the IEEE802.11 standards but maintained and governed by the non-profit Wi-Fi Alliance.

802.11 owes its creation to NCR in 1991 who first developed the wireless protocol as a means of networking cash registers. It wasn't until 1999 when the Wi-Fi Alliance was formed that the technology became ubiquitous and pervasive in the burgeoning PC and notebook market. The original protocol is vastly different than modern 802.11 b/g/n/ac protocols. It only supported a 2 Mbps data rate with forward error correction.

The success of IEEE802.11 can be attributed to the layered stack approach of the OSI model. Simply replacing the MAC and PHY layers with IEEE802.11 layers allowed existing TCP/IP infrastructure to seamless be used. Today, nearly every mobile device, notebook, tablet, embedded system, toy, and video game incorporate an IEEE802.11 radio of some kind. That said, 802.11 has had a storied past, particularly in the security model. The original 802.11 security model was based on the UC Berkeley Wired Equivalent Privacy security mechanism that was later proven to be unreliable and easily compromised. Several notable exploits including the TJ Maxx data breach through 802.11 WEP in 2007 resulted in 45 million stolen credit cards. Today, Wi-Fi Protected Access (WPA) and WPA2 using AES 256-bit pre-shared keys have certainly tightened up security and WEP is rarely used.

This section will detail some of the differences in the 802.11 protocols and particular information relevant to the IoT architect. We will detail the current IEEE802.11ac design and then examine 802.11ah HaLow and 802.11p V2V, as all three are relevant to the internet of things.

IEEE 802.11 suite of protocols and comparison

The IEEE LAN/MAN Standards Committee maintains and governs the IEEE 802 specification(s). The original 802.11 goal was to provide a link layer protocol for wireless networking. This evolved from the 802.11 base specification to 802.11ac in 2013. Since then, the working group has focused on other areas, as the following chart illustrates. Specific 802.11 variants have been examined for use cases and segments such as low power/low bandwidth IoT interconnect (802.11ah), vehicle-to-vehicle communication (802.11p), reuse of television analog RF space (802.11af), extreme bandwidth near meter communication for audio/video (802.11ad), and of course the follow-on to the 802.11ac standard (802.11ax).

The new variants are designed for different areas of the RF spectrum or to reduce latency and improve safety for vehicular emergencies. The following table should reflect the tradeoffs between range, frequency, and power. We will cover aspects in the chart such as modulation, MIMO streams, and frequency usage later in this section.

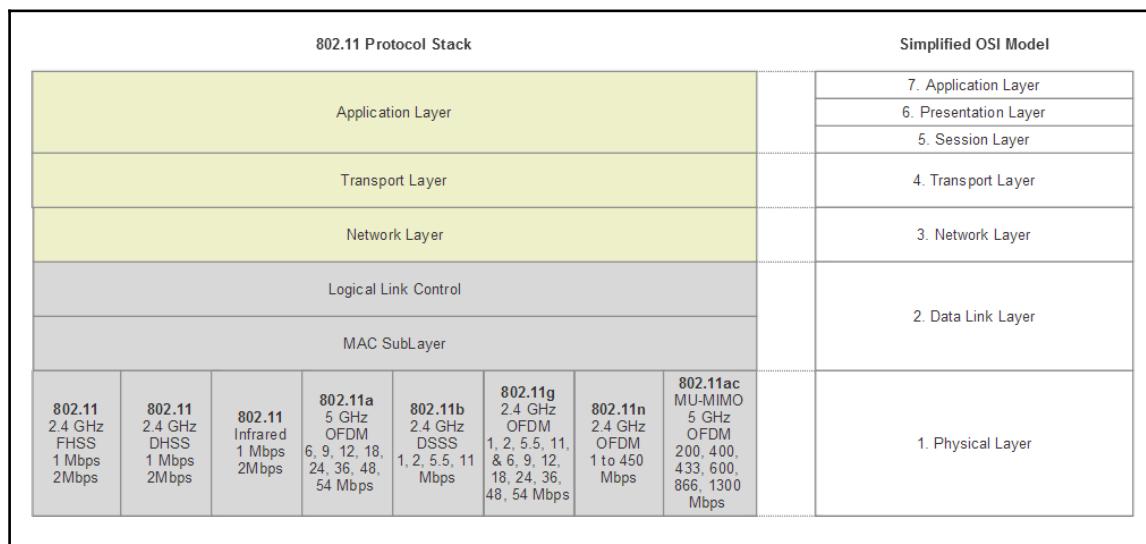
IEEE 802.11 Protocol	Use Case	Release Date	Frequency (GHz)	Bandwidth (MHz)	Streaming Data Rate per Channel min-max (Mbps)	Allowable MIMO Streams	Modulation	Indoor Range (m)	Outdoor Range (m)	Typical Dissipated Power per Chip (mW)
802.11	First 802.11 design	Jun-97	2.4	22	1 to 2	1	DSSS, FHSS	20	20	50
a	Release simultaneously with 802.11b Less prone to interference than 802.11b	Sep-99	5	20	6 to 54	1	OFDM	30	120 5000	50
			3.7				[SISO]			
b	Release simultaneously with 802.11a Significant speed increase over 802.11a at improved range	Sep-99	2.4	22	1 to 11	1	DSSS (SISO)	50	150	7 to 50
g	Speed increase over 802.11b	Jun-03	2.4	20	6 to 54	1	OFDM, DSSS (SISO)	38	140	50
n	Multiple antenna technology for improved speed, and range.	Oct-09	2.4 / 5	20	7.2 to 72.2	4	OFDM (MIMO)	70	250	40
				40	15 to 150					
ac	Better performance and coverage over 802.11n. Wider channel and improved modulation. Allows multiple users using MU-MIMO. Introduced beamforming.	Dec-13	5	20	7.2 to 96.3	8	OFDM (MU_MIMO)	35	35	40
				40	15 to 200					
				80	32.5 to 433.3					
				160	65 to 866.7					
ah	"WiFi HaLow"	Dec-16	2.4 / 5	1 to 16	347	4	OFDM	1000	1000	tbd but goal is low power
p	"Wireless Access in Vehicular Environments" "Intelligent Transport Systems" Dedicated Short Range Communication Trasnport uses cases: toll collection, safety and collision emergencies, vehicular networking.	Jun-09	5.9	10	27	1	OFDM	NA	400 to 1000	40
af	"white WiFi" or "Super WiFi" Deploy unused spectrum in TV bands to provide last mile connectivity in India, Kenya, Singapore, US and UK	Nov-13	0.470 to 0.710	6 to 8	568	4	OFDM	NA	6000-100,000	tbd
ad	WiGig Alliance 60 GHz Wireless for HD video and projectors Audio and video transport and cable replacement	Dec-12	60	2160	4260	>10	SC_OFDM (MU-MIMO)	10	10	tbd
ax	"High Efficiency Wireless (HEW)" Next gen 802.11 4x increase in capacity over 802.11ac Average increase of 4x speed per user over 802.11ac Backwards compatible to 802.11a/b/g/n/ac Dense deployment scenarios	2019	2.4 / 5	20	450 to 10000	8	OFDMA (MU-MIMO)	35	35	tbd
				40						
				80						
				160						

Various IEEE802.11 standards and specifications from the legacy 802.11 original specifications to the yet to be ratified 802.11ax.

IEEE 802.11 architecture

The 802.11 protocol represents a family of wireless radio communications based on different modulation techniques in the 2.4 GHz and 5 GHz ISM bands of the unlicensed spectrum. 802.11b and 802.11g reside in the 2.4 GHz band, while 802.11n and 802.11ac open up the 5 GHz band. The last chapter detailed the 2.4 GHz band and the different protocols that reside in that space. Wi-Fi is susceptible to the same noise and interference as Bluetooth and Zigbee and deploys a variety of techniques to ensure robustness and resiliency.

From a stack perspective, the 802.11 protocols reside in the link layer (one and two) of the OSI model, as shown in the following figure:



IEEE 802.11ac Stack.

The stack includes various PHYs from older 802.11 specifications such as the 802.11 original PHYs (including infrared), a, b, g, and n. This is to ensure backward compatibility across networks. Most chipsets include the entire PHY collection and it is difficult to find a part with an older PHY alone.

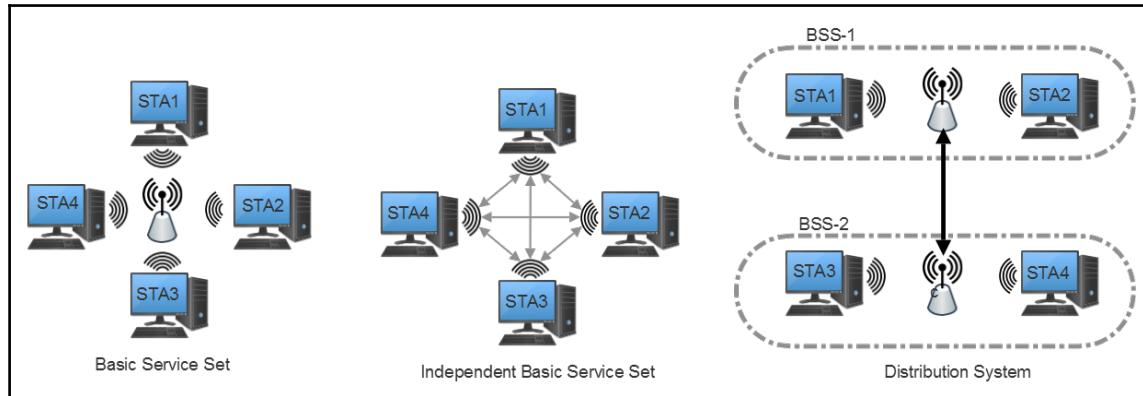
802.11 systems support three basic topologies:

- **Infrastructure:** In this form, a **Station (STA)** refers to an 802.11 endpoint device (like a Smartphone) that communicates with a central **access point (AP)**. An AP can be a gateway to other networks (WAN), a router, or a true access point in a larger network. This is also known as **Infrastructure Basic Set Service (BSS)**. This topology is a star topology.
- **Ad hoc:** 802.11 nodes can form what is called an **Independent Basic Set Service (IBSS)** where each station communicates and manages the interface to other stations. No access point or a star topology is used in this configuration. This is a peer-to-peer type of topology.
- **Distribution system (DS):** The DS combines two or more independent BSS networks through access point interconnects.

Note: IEEE802.11ah and IEEE802.11s support forms of a mesh topology.



Below are examples of the three basic topologies of an IEEE 802.11 architecture:

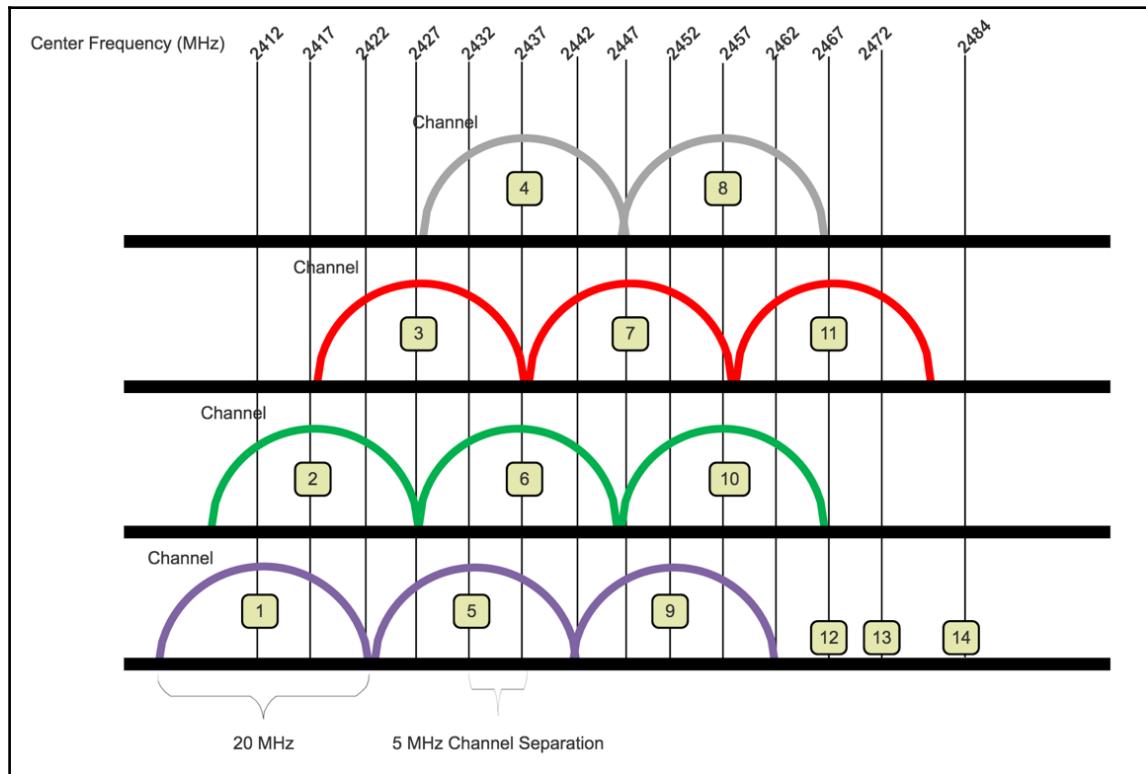


802.11 network architectures. BSS, IBSS, and a Distribution System combine two independent BSS.

In total, the 802.11 protocol allows for up to 2007 STAs to be associated with a single access point. This is relevant when we explore other protocols, such as the IEEE 802.11ah for IoT, later in this chapter.

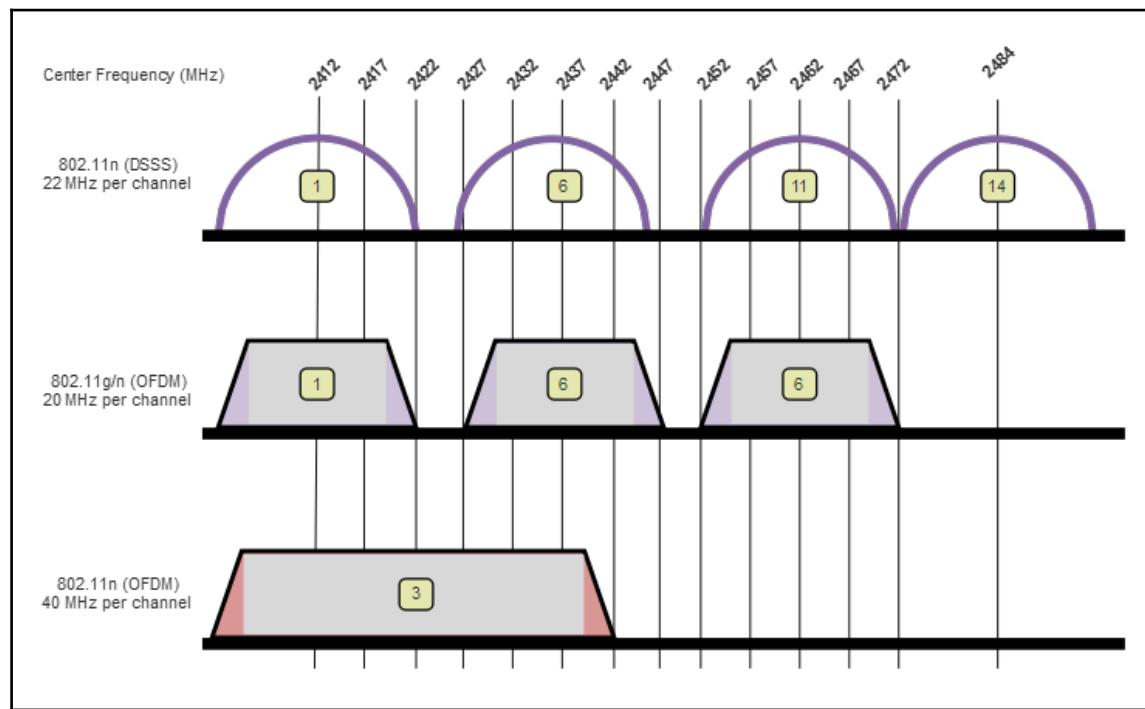
IEEE 802.11 spectrum allocation

The first 802.11 protocol used a spectrum in the 2 GHz and 5 GHz ISM region and evenly spaced channels roughly 20 MHz apart from each other. The channel bandwidth was 20 MHz, but later amendments from IEEE allowed 5 MHz and 10 MHz to operate as well. In the United States, 802.11b and g allow for eleven channels (other countries may support up to fourteen). The following figure depicts the channel separation. Three of the channels are non-overlapping (1,6,11):



802.11 2.4GHz frequency space and non-interfering channel combinations. Note the 5 MHz channel separation across 14 channels each 20 MHz wide.

802.11 specifies a spectral mask, which defines the permitted power distribution across each channel. The spectral mask requires the signal be attenuated to certain levels (from its peak amplitude) at specified frequency offsets. That said, signals will tend to radiate into adjacent channels. 802.11b using **Direct Sequence Spread Spectrum (DSSS)** has a completely different spectral mask than 802.11n using **Orthogonal Frequency Divisional Multiplexing (OFDM)**. OFDM has a much denser spectral efficiency and therefore also sustains much higher bandwidth. Shown below are the channel and modulation differences between 802.11 b, g and n. Channel width limits the number of simultaneous channels from 4 to 3 to 1. The shape of the signal also varies between DSSS and OFDM. OFDM being much denser and thus capable of higher bandwidth.



Differences between 802.11b,g, and n using DSSS versus OFDM and carrying channel widths.



While there are 14 channels in the 2.4 GHz range, the usage of the channels is governed by region and country. For example, North America allows the usage of channels 1 through 11, Japan allows all 14 channels for 802.11b and 1 through 13 for 802.11g/n, Spain allows only channels 10 and 11, and France allows 10 through 13. The allocations vary and designers should be aware of country restrictions. IEEE uses the nomenclature *regdomain* to describe country channel, power, and time restrictions that impact the PHY.

IEEE 802.11 modulation and encoding techniques

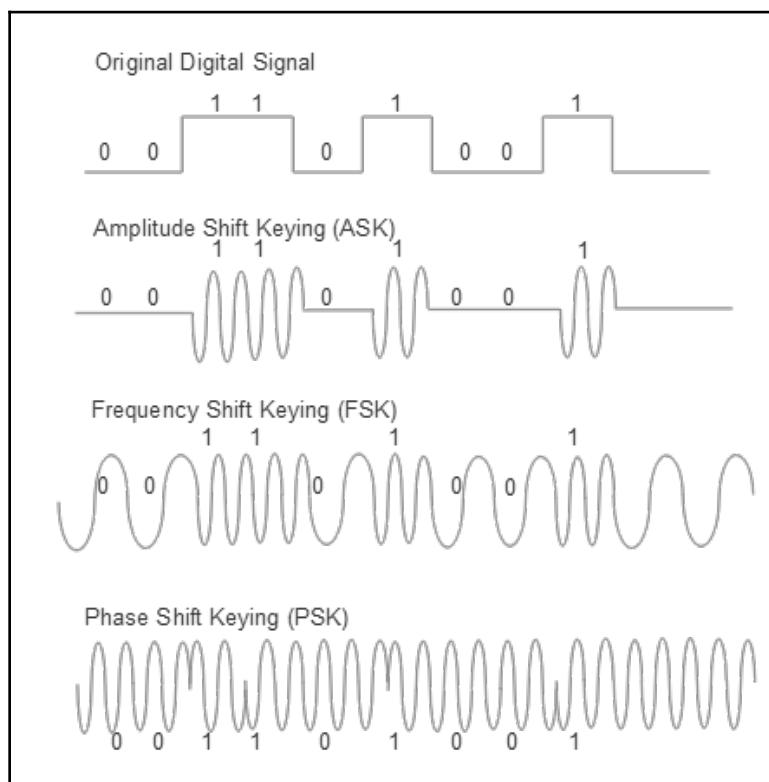
This section details the techniques of modulation and encoding in IEEE 802.11 protocol. These techniques are not unique to 802.11; they also apply to 802.15 protocols and, as we will see, to cellular protocols as well. The methods of frequency hopping, modulation, and phase-shift keying are fundamental methods that should be understood by an architect as different techniques balance range, interference, and throughput.

Digital data transmitted by an RF signal must be transformed into analog. This occurs at the PHY no matter what RF signal is being described (Bluetooth, Zigbee, 802.11, and so on). An analog carrier signal will be modulated by a discrete digital signal. This forms what is called a symbol or a modulation alphabet. A simple way to think about symbol modulation is a piano with four keys. Each key represents two bits (00, 01, 10, 11). If you can play 100 keys per second, that implies you can transmit 100 symbols per second. If each symbol (tone from the piano) represents two bits, then it is equivalent to a 200 bps modulator. While there are many forms of symbol encoding to study, the three basic forms include:

- **Amplitude Shift Keying (ASK):** This is a form of amplitude modulation. Binary 0 is represented by one form of modulation amplitude and 1 a different amplitude. A simple form is shown in the following figure, but more advanced forms can represent data in groups using additional amplitude levels.
- **Frequency Shift Keying (FSK):** This modulation technique modulates a carrier frequency to represent 0 or 1. The simplest form shown in the following figure is **Binary Frequency Shift Keying (BPSK)**, which is the form used in 802.11 and other protocols. In the last chapter, we talked about Bluetooth and Z-Wave, those protocols use a form of FSK called **Gaussian Frequency Shift Keying (GFSK)** that filters the data through a Gaussian filter, which smooths the digital pulse (-1 or +1) and shapes it to limit spectral width.

- **Phase Shift Keying (PSK):** Modulates the phase of a reference signal (carrier signal). Used primarily in 802.11b, Bluetooth, and RFID tags. PSK uses a finite number of symbols represented as different phase changes. Each phase encodes an equal number of bits. A pattern of bits will form a symbol. The receiver will need a contrasting reference signal and calculate the difference to extract the symbols and then demodulate the data. An alternative method requires no reference signal for the receiver. The receiver will inspect the signal and determine if there is a phase change without referencing a secondary signal. This is called **Differential Phase Shift Keying (DPSK)** and is used in 802.11b.

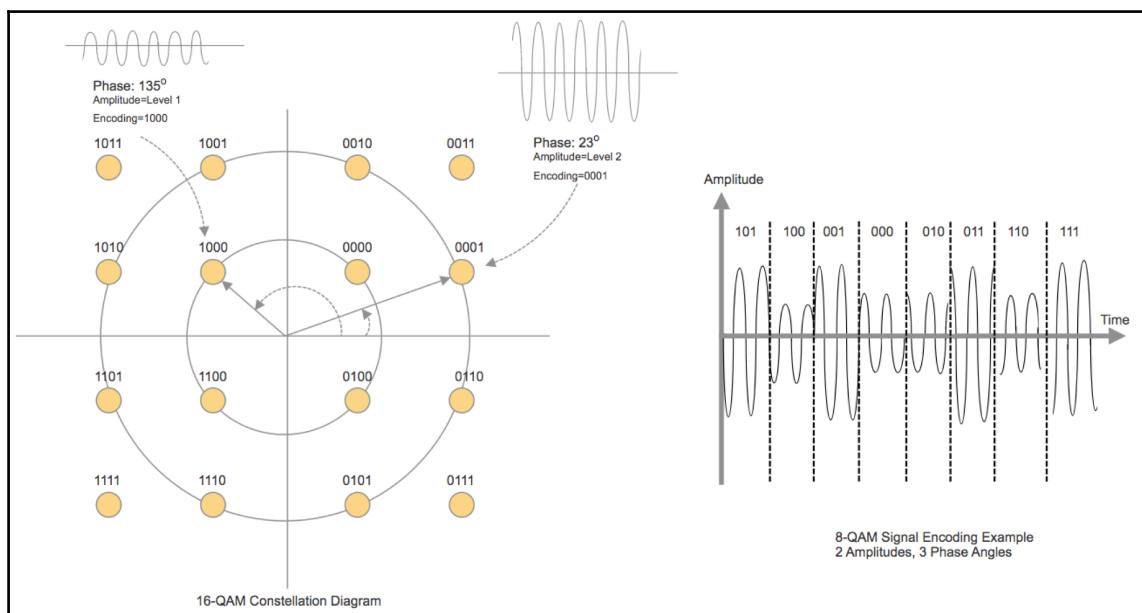
The following figure pictographically depicts the various encoding methods:



Different forms of symbol encoding using keying techniques: amplitude keying, frequency keying, and phase keying. Note how phase keying changes phase for every "1" encountered.

The next form of modulation technique is hierarchical modulation, specifically **Quadrature Amplitude Modulation (QAM)**. The following constellation diagram represents the encoding in a 2D Cartesian system. The length of any one vector represents the amplitude, and the angle to a constellation point represents the phase. Generally speaking, there are more phases than amplitudes that can be encoded, as shown in the following 16-QAM constellation diagram. The 16-QAM has three amplitude levels and 12 total phases. This allows for 16 bits to be encoded. 802.11a and 802.11g can use 16-QAM and even higher density 64-QAM. Obviously, the denser the constellation, the more encoding can be represented and the higher the throughput.

The figure shown illustrates a QAM encoding process pictographically. On the left is a representation of a 16 point (16-QAM) constellation diagram. There are 3 amplitude levels indicated by the length of the vectors and 3 phases per quadrant indicated by the angle of the vector. This allows for 16 symbols to be generated. These symbols are reflected in varying the phase and amplitude of the signal generated. On the right is an 8-QAM example waveform diagram showing the varying phases and amplitudes that represent a 3 bit (8 value) modulation alphabet.



Quadrature Amplitude Modulation (QAM). Left: 16-QAM Constellation. Right: 8-QAM Waveform Encoding.



QAM has practical limits. Later, we will see very dense constellations that radically increase throughput, one can only add a certain amount of phase angles and amplitudes. Noise generated by **Analog to Digital Converters (ADC)** and **Digital to Analog Converters (DAC)** will introduce quantization errors and noise and will need to sample signals at very high speeds. Additionally, the **Signal to Noise Ratio (SNR)** must exceed a certain value to achieve a good **Bit Error Rate (BER)**.

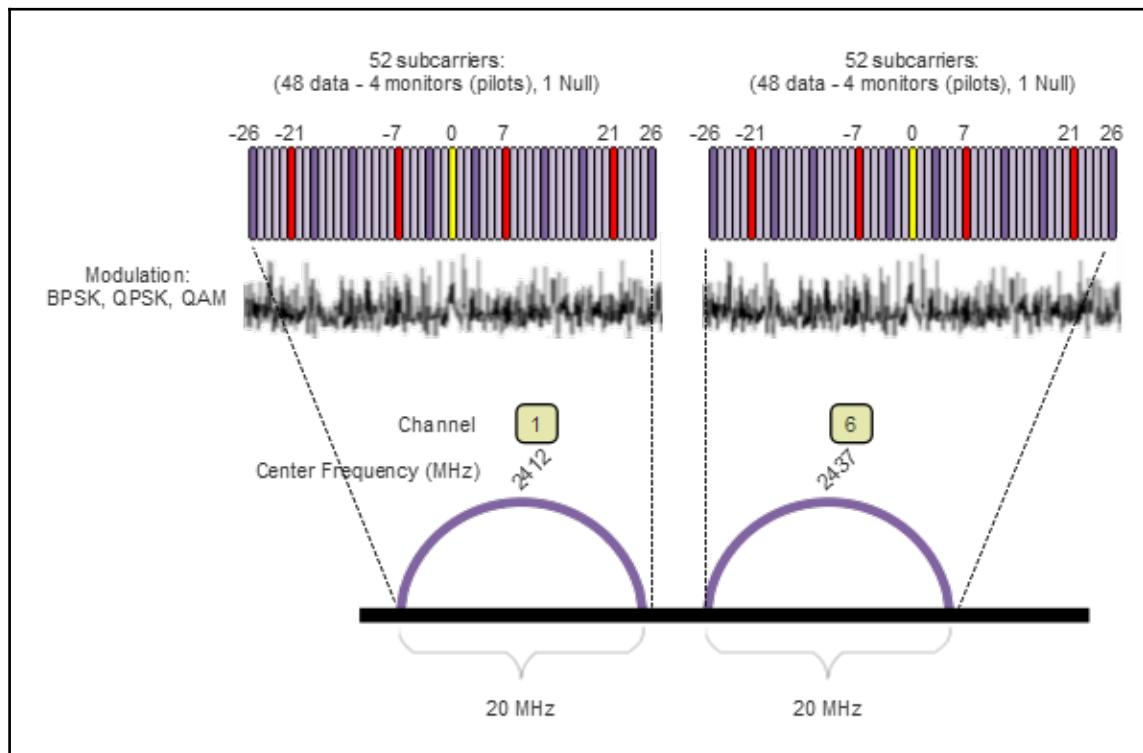
The 802.11 standards employ different interference mitigation techniques that essentially spread a signal across a band:

- **Frequency Hopping Spread Spectrum (FHSS):** Spreads signal over 79 non-overlapping channels that are 1 MHz wide in the 2.4 GHz ISM band. Uses a pseudo-random number generator to start the hopping process. Dwell Time refers to the minimum time a channel is used before hopping (400 ms). Frequency hopping was also described in the last chapter and is a typical scheme for spreading signals.
- **Direct sequence spread spectrum:** First used in 802.11b protocols and has 22 MHz-wide channels. Each bit is represented by multiple bits in the signal transmitted. The data being transmitted is multiplied by a noise generator. This will effectively spread the signal over the entire spectrum evenly using a pseudo-random number sequence (called the *Pseudo-Noise PN code*). Each bit is transmitted with an 11-bit *chipping sequence* (phase shift keying). The resulting signal is an XOR of the bit and the 11-bit random sequence. DSSS delivers about 11 million symbols per second when we consider the chipping rate.
- **OFDM:** Used in IEEE 802.11a and the newer protocols. This technique divides a single 20 MHz channel into 52 sub-channels (48 for data and four for synchronization and monitoring) to encode data using QAM and PSM. A **Fast Fourier Transform (FFT)** is used to generate each OFDM symbol. A set of redundant data surrounds each sub-channel. This redundant band of data is called the **Guard Interval (GI)** and is used to prevent **Inter-symbol Interference (ISI)** between neighbor subcarriers. Notice the subcarriers are very narrow and have no guard bands for signal protection. This was intentional because each subcarrier is spaced equally to the reciprocal of the symbol time. That is, all the sub-carriers transport a complete number of sine-wave cycles that when demodulated will sum to zero. Because of this, the design is simple and doesn't need the extra cost of bandpass filters. IEEE 802.11a uses 250,000 symbols per second. OFDM is generally more efficient and dense (hence more bandwidth) than DSSS and is used in the newer protocols.



Fewer symbols per second have an advantage in situations where there are reflections of signals on walls and windows. Since the reflections will cause what is called multi-path distortion (copies of the symbol hit the receiver at different times), a slower symbol rate allows more time to transmit a symbol and there is more resilience to delay spread. However, if a device is moving, there can be Doppler effects that affect OFDM more than DSSS. Other protocols such as Bluetooth use one million symbols per second.

The following figure depicts an OFDM system with 52 subcarriers in two 20-MHz channels:



An example of OFDM. Here a channel is subdivided into 52 smaller slots or subcarriers (each carrying a symbol).

The set of different modulations available for each standard is called the **Modulation and Coding Scheme (MCS)**. The MCS is a table of available modulation types, guard intervals, and coding rates. One references this table with an index.



Since 802.11b entered the market, before 802.11a they used different encoding schemes. Each is incompatible with each other. There is some confusion in the market as to the difference as protocols were released nearly simultaneously.

IEEE 802.11 MIMO

MIMO is the acronym that refers to multiple input multiple output. MIMO exploits a previously mentioned RF phenomenon called multipath. Multipath transmission implies that signals will reflect off walls, doors, windows, and other obstructions. A receiver will see many signals each arriving at different times via different paths. Multipath tends to distort signals and cause interference, which eventually degrades signal quality (this effect is called multipath fading). With the addition of multiple antennas, a MIMO system can linearly increase the capacity of a given channel by simply adding more antennas. There are two forms of MIMO:

- **Spatial diversity:** This refers to transmit-and-receive diversity. A single stream of data is transmitted on multiple antennas simultaneously using space-time coding. These provide improvements in the signal to noise ratio and they are characterized by their improvement of link reliability and coverage of the system.
- **Spatial multiplexing:** Used to provide additional data capacity by utilizing the multiple paths to carry additional traffic; that is, increasing the data throughput capability. Essentially, a single high-rate data stream will be split into multiple separate transmissions on different antennas.

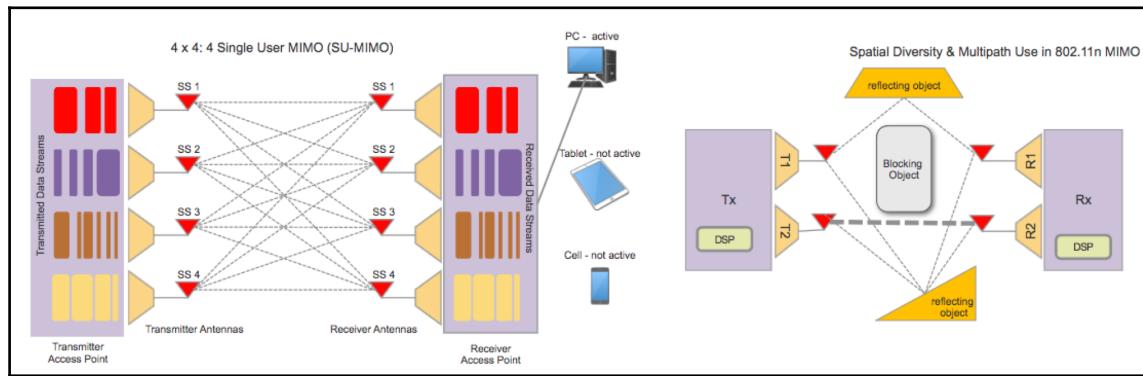


In the previous chapter, we introduced frequency hopping techniques in PAN networks such as Bluetooth. Frequency hopping is one method to overcome the multipath fading problem by constantly changing the angles of the multipath. This has the effect of distorting the RF signal size. Bluetooth systems typically have one antenna, thereby making MIMO difficult. As far as Wi-Fi is concerned, only the original 802.11 standard supported a form of frequency hopping (FHSS). OFDM systems maintain a channel lock, and thus can be subject to the multipath fading problem.

Using multiple streams does impact overall power usage. IEEE 802.11n includes a mode to only enable MIMO when the effect will have a performance benefit, thus saving power at all other times. The Wi-Fi Alliance requires all products to support at least two spatial streams to receive 802.11n compliance.

A WLAN will split data into multiple streams called spatial streams. Each transmitted spatial stream will use a different antenna on the transmitter. IEEE 802.11n allows for four antennas and four spatial streams. By using multiple streams sent separately for antennas spaced apart from each other, spatial diversity in 802.11n gives some confidences that at least one signal will be strong enough to reach the receiver. At least two antennas are needed to support MIMO functionality. Streaming is also modulation agnostic. BPSK, QAM, and other forms of modulation work with spatial streaming. A digital signal processor on the transmitter and receiver will adjust for the multipath effects and delay the line of sight transmission by just enough time to have it line up perfectly with the non-line-of-sight paths. This will cause the signals to reinforce.

The IEEE 802.11n protocol supports a four-stream single user MIMO (SU-MIMO) implementation, meaning that the transmitters would work in unison to communicate to a single receiver. Here 4 transmit and 4 receive antennas deliver multiple streams of data to a single client. Right: the effect of spatial diversity MIMO in 802.11n. Following is an illustration of SU-MIMO and multipath use in 802.11n.



Left: Illustration of SU-MIMO in IEEE 802.11n. Right: the effect of spatial diversity MIMO in 802.11n.

In the figure, four transmit and four receive antennas deliver multiple streams of data to a single client (SU-MIMO). On the right, two transmitters spaced a fixed distance apart communicate to two receivers. Multiple paths exist as reflections from the two transmitters. One line-of-sight path is stronger and will be favored. DSPs on the transmitters and receiver side also mitigate multipath fading by combining signals so the resulting signal exhibits little fading.

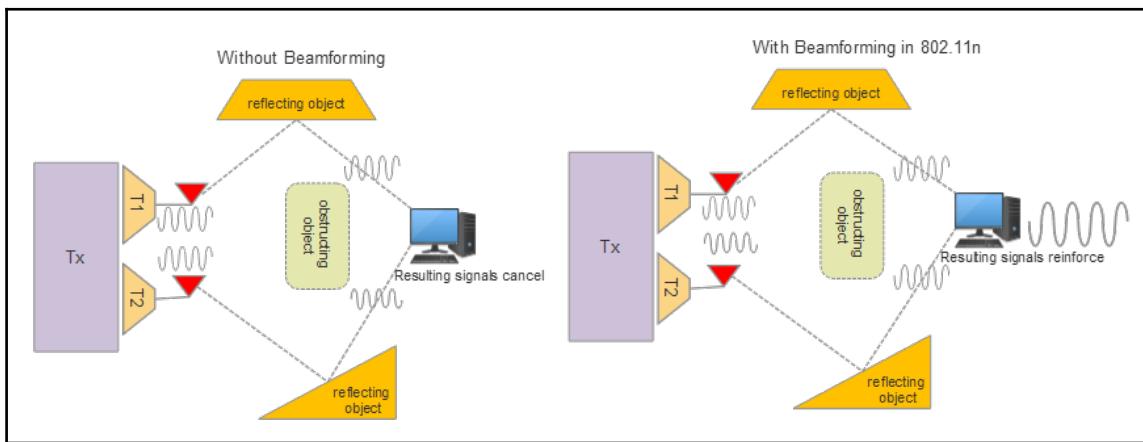


The IEEE 802.11 protocol identifies MIMO streams by the notation $M \times N : Z$, where M is the maximum number of transmit antennas and N is the maximum number of receiver antennas. Z is the maximum number of data streams that can be used simultaneously. So, a MIMO of $3 \times 2 : 2$ implies there are three transmit stream antennas and two receive stream antennas, but can only send or receive two simultaneous streams.

802.11n also introduced the optional feature of beamforming. 802.11.n defines two types of beamforming methods: implicit feedback and explicit feedback:

- **Implicit feedback beamforming:** This mode assumes that the channel between the beamformer (AP) and the beamformee (client) are the reciprocal (same quality in both directions). If that is true, the beamformer transmits a training request frame and receives a sounding packet. With the sounding packet, the beamformer can estimate the receiver's channel and builds a steering matrix.
- **Explicit feedback beamforming:** In this mode, the beamformee responds to a training request by computing its own steering matrix, and sends back the matrix to the beamformer. This is a more reliable method.

Below is a diagram illustrating the effects of beamforming in a situation with no line-of-sight communication. In the worst case, the signals arrive 180 degrees out of phase and nullify each other. With beamforming, the signals can be adjusted in phase to strengthen each other at the receiver.



Examples of a system with and without beamforming. In this case, a system has no direct line of sight and relies on reflections to propagate a signal.

Beamforming relies on multiple spaced antennas to focus a signal at a particular location. The signals can be adjusted in phase and magnitude to arrive at the same location and reinforce each other providing better signal strength and range. Unfortunately, 802.11n did not standardize on a single method for beamforming and left that up to the implementor. Different manufacturers used different processes and could only guarantee it would work with identical hardware. Thus, beamforming was not adopted heavily in the 802.11n timeframe.

We will cover MIMO technologies in many other areas, such as 802.11ac, and in the next chapter on long-range communication using cellular 4G LTE radios.

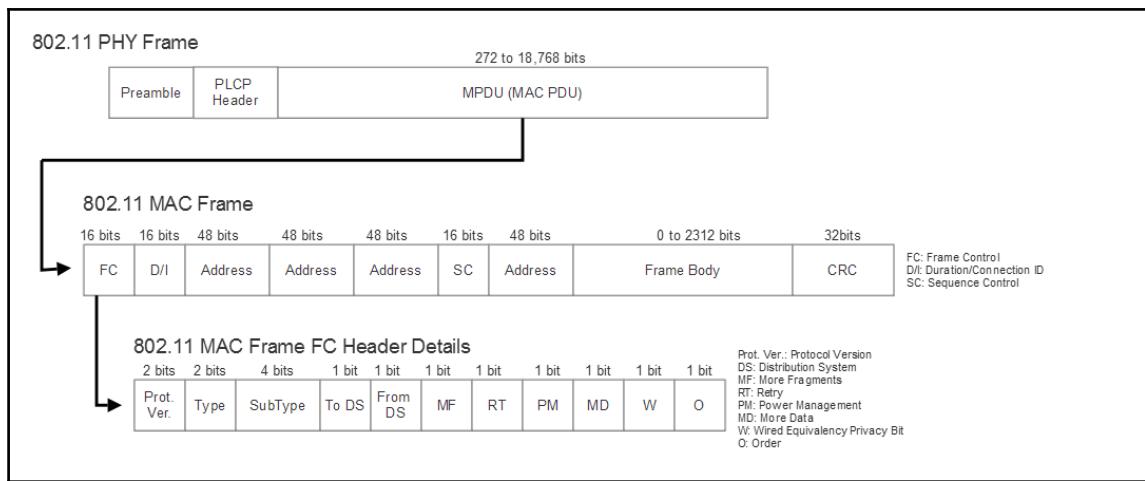
IEEE 802.11 packet structure

802.11 uses the typical packet structure we have seen before with headers, payload data, frame identifiers, and so on. Starting with the PHY frame organization, we have three fields: a preamble, which assists in the synchronization phase, a PLCP header, which describes the packet configuration and characteristics such as data rates, and the MPDC MAC data.

Each IEEE 802.11 specification has a unique preamble and is structured by the number of symbols (described later) and not by the number of bits for each field. Examples of the preamble structures are as follows:

- **802.11 a/g:** Preamble includes a short training field (two symbols) and a long training field (two symbols). These are used by the subcarriers for timing sync and frequency estimation. Additionally, the preamble includes a signal field that describes the data rate, length, and parity. The signal determines how much data is being transmitted in that particular frame.
- **802.11 b:** Preamble will use either a long sequence of 144 bits or a short sequence of 72 bits. The header will include signal rate, service modes, length of data in microseconds, and a CRC.
- **802.11n:** Has two operating modes: Greenfield (HT) and mixed (non-HT). Greenfield can only be used where no legacy systems exist. Non-HT mode is a compatibility mode with 802.11a/g systems and delivers no better performance than a/g. Greenfield mode allows for higher speed transport.

The following illustration is the 802.11 PHY and link layer packet frame structure:



802.11 Generalized PHY and MAC frame structure

The MAC frame structure is shown in the preceding figure. The MAC frame contains the plurality of representative fields. The frame control (FC field) subfields are detailed as follows:

- **Protocol version:** Indicates version of the protocol used.
- **Type:** WLAN frame as control, data, or management frame type.
- **Subtype:** Further delineation of frame type.
- **ToDS and FromDS:** Data frames will set one of these bits to 1 to indicate if the frame is headed to a distribution system. IBSS ad hoc network.
- **More fragments:** If a packet is divided into many frames, then every frame except the last will have this bit-set.
- **Retry:** Indicates a frame was resent and assists in resolving duplicate frames being transmitted.
- **Power management:** Indicates the power state of the sender. APs cannot set this bit.
- **More data:** An AP will use this bit to assist when STAs are in a power save mode. This bit is used to buffer frames in a distribution system.
- **Wired equivalent privacy:** Set to a 1 when a frame is decrypted.
- **Order:** If a strict order mode is used in the network this bit will be set. Frames may not be sent in-order and strict order mode forces in-order transmission.

Moving up the MAC frame from the frame control field, we first examine the duration/connection ID bit:

- **Duration/connection ID:** Indicates duration, contention-free period, and association ID. The association ID is registered during Wi-Fi initial handshaking.
- **Address fields:** 802.11 can manage four MAC addresses in the following order:
 - **Address 1:** Receiver
 - **Address 2:** Transmitter
 - **Address 3:** Used for filtering
- **SC:** Sequence control is a 16-bit field for message order.

The 802.11 protocol has several types of frames represented by the type and subtype fields. There are three fundamental types: management frames, control frames, and data frames.

Management frames provide network administration, security, and maintenance. The following table defines the types of management frames:

Frame name	Description
Authentication frame	An STA will send an authentication frame to an AP, which responds with its own authentication frame. Here, the shared key is sent and verified using a challenge response.
Association request frame	This is transmitted from an STA to request an AP to synchronize. It contains the SSID the STA wants to join and other information for synchronization.
Association response frame	Transmitted from an AP to a STA contain and acceptance or rejection message to an association request. If accepted, an association ID will be sent in the payload.
Beacon frame	This is the periodic beacon broadcast from an AP. Includes the SSID.
Deauthentication frame	Transmitted from a STA wishing to leave a connection from another STA.
Disassociation frame	Transmitted from a STA wishing to terminate a connection.
Probe request frame	Broadcast from an STA to another STA.
Probe response frame	Transmitted from an AP in response to a probe request. Contains information such as supported data rates.

Reassociation frame	Used when an STA loses signal strength with one AP but finds another AP associated with the network using a stronger signal. The new AP will attempt to associate with the STA and forward information stored in the original AP buffer.
Reassociation response frame	Transmitted from the AP with acceptance or rejection to a reassociation request.

The next major frame type is the control frame. Control frames help exchange data between STAs:

Frame name	Description
Acknowledgement frame (ACK)	A receiving STA will always ACK received data if no errors have occurred. If the sender does not receive an ACK after a fixed time, the sender will resend the frame.
Request to rend frame (RTS)	This is part of the collision avoidance mechanism. An STA will begin by sending an RTS message if it wishes to transmit some data.
Clear to send frame (CTS)	STA response to an RTS frame. Request STA can now send the data frame. This is a form of collision management. A time value is used to hold off transmissions from other STAs with the requesting STA transmits.

The final frame type is the data frame. This is the bulk of the data-carrying function of the protocol.

IEEE 802.11 operation

As mentioned previously, an STA is considered a device equipped with a wireless network interface controller. An STA will always be listening for active communication in a specific channel. The first phase of connecting to Wi-Fi is the scanning phase. There are two types of scan mechanisms employed:

- **Passive scanning:** This form of scanning uses beacons and probe requests. After a channel is selected, the device performing the scan will receive beacons and probe requests from nearby STAs. An access point may transmit a beacon, and if the STA receives the transmission, it may progress to join the network.

- **Active scanning:** In this mode, the STA will attempt to locate an access point by instantiating probe requests. This mode of scanning uses more power but allows for faster network joins. The AP may respond to a probe request with a probe request response, which is similar to a beacon message.



An access point will typically broadcast a beacon at a fixed time interval called the **Target Beacon Transmit Time (TBTT)**. TBTT is typically once every 100 ms.

Beacons are always broadcast at the lowest basic rates to ensure every STA in the range has the ability to receive the beacon even if it can't connect to that particular network. After a beacon is processed, the next phase of Wi-Fi connectivity is the synchronization phase. This phase is necessary to keep clients attuned to the access point. The beacon packet contains information needed by the STA:

- **SSID:** Service Set ID. 1 to 32-character network name (this field can optionally be hidden by setting the SSID length to zero. Even if it is hidden the other portions of the beacon frame are transmitted as usual. Generally, speaking, using a hidden SSID offers no additional network security.
- **BSSID:** Basic Service Set ID. Unique 48-bit following layer-2 MAC address conventions. Formed by the combination of the 24-bit Organization Unique Identifier and the manufacturer's assigned 24-bit identifier for the radio chipset.
- **Channel width:** 20 MHz, 40 MHz, and so on
- **Country:** List of supported channels (country-specific)
- **Beacon interval:** TBTT time mentioned previously
- **TIM/DTIM:** Wake-up times and intervals to retrieve broadcast messages—allows for advanced power management.
- **Security services:** WEP, WPA, and WPA2 abilities



Beacons are an interesting concept with a likeness to Bluetooth beacons. Bluetooth wireless offers much greater message capabilities and flexibility in beacon broadcasts, but there is a range of products and services that make use of Wi-Fi beaconing as well.

If the STA does find an AP or another STA to make a connection with, it then enters an authentication phase. More will be discussed on the variety of security standards used in 802.11 later in this chapter.

If the security and authentication process succeeds, the next phase is association. The device will send an association request frame to the AP. The AP will subsequently reply with an association response frame that will allow the STA to join the network or be excluded. If the STA is included, the AP will release an association ID to the client and add it to the list of connected clients.

At this point, data can be exchanged with the AP and vice versa. All data frames will be followed by an acknowledgment.

IEEE 802.11 security

In the previous section, we described the association process for a Wi-Fi device to join a network. One of the phases involved was authentication. This section will cover the various types of authentication used on Wi-Fi WLANs and various strengths and weaknesses:

- **WEP:** Wired equivalent privacy. This mode sends a key in plain text from the client. The key is then encrypted and sent back to the client. WEP uses different size keys but they are typically 128 bit or 256 bit. WEP uses a shared key, which means that the same key is available to all clients. It can be easily compromised by simply listening and sniffing for all the authentication frames coming back to clients joining a network to determine the key used for everyone. Due to a weakness in the key generation, the first few bytes of the pseudo-random string may reveal (5% probability) a portion of the key. By intercepting five to 10 million packets, an attacker could with reasonable confidence get enough information to reveal the key.
- **WPA:** Wi-Fi protected access (or WPA-Enterprise) was developed as the IEEE 802.11i security standard to replace WEP and is a software/firmware solution requiring no new hardware. One significant difference is WPA uses a **Temporal Key Integrity Protocol (TKIP)**, which performs per-packet key mixing and rekeying. This means that each packet will use a different key to encrypt itself, unlike in the case of WEP. WPA starts by generating a session key based on the MAC address, temporal session key, and the initialization vector. This is fairly processor-intensive but only performed once per session. The next step is to extract the lowest 16 bits of a received packet with the result of the bits generated in phase one. This is the 104-bit per-packet key. Data can now be encrypted.
- **WPA-PSK:** WPA pre-shared key or WPA-Personal. This mode exists where there is no 802.11 authentication infrastructure. Here, one uses a passphrase as a pre-shared key. Each STA can have their own pre-shared key associated with its MAC address. This is similar to WEP, and weaknesses have already been found if the pre-shared key uses a weak passphrase.

- **WPA2:** This replaces the original WPA design. WPA2 uses AES for encryption, which is much stronger than TKIP in WPA. This encryption is also called CTR mode with CBC-MAC Protocol, or CCMP for short.



To achieve high bandwidth rates in 802.11n, CCMP mode must be used or the data rate will not exceed 54 Mbps. Additionally, WPA2 certification is needed to use the Wi-Fi Alliance trademark logo.

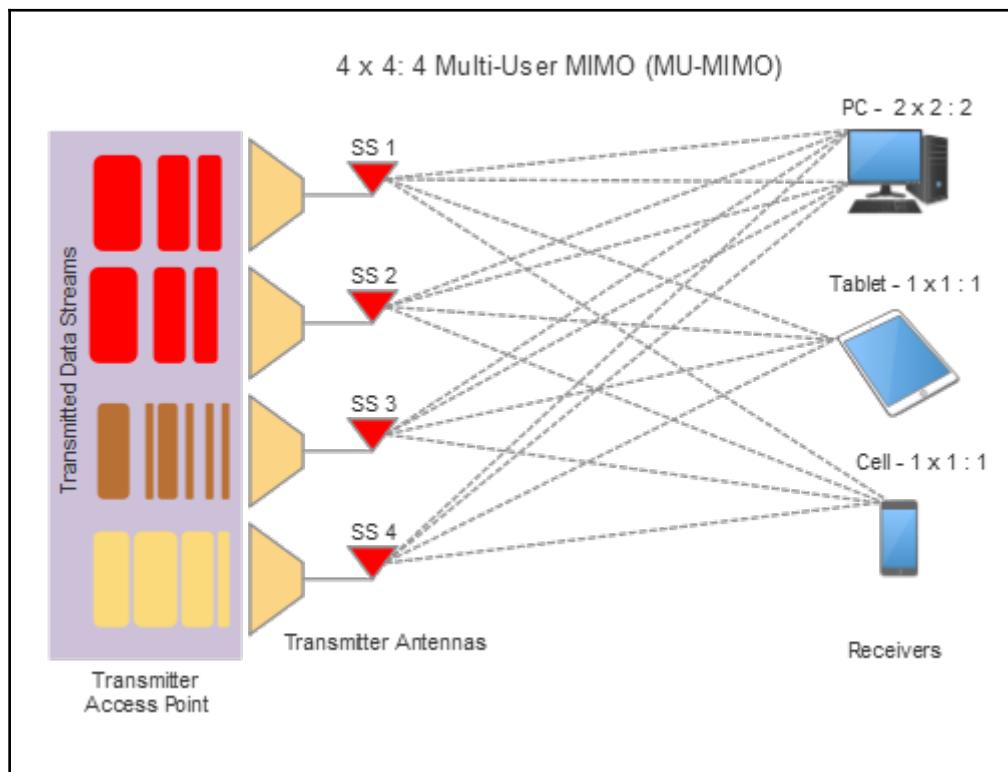
IEEE 802.11ac

IEEE802.11ac is the next generation WLAN and follow-on to the family of 802.11 standards. IEEE802.11ac was approved as a standard in December of 2013 after five years of work. The goal is to deliver multi-station throughput at least 1 GBps and a single link throughput of 500 Mbps. The technology accomplishes this through wider channel bandwidth (160 MHz), more MIMO spatial streams, and extreme density modulation (256-QAM). 802.11ac exists only in the 5 GHz band, yet will coexist with previous standards (IEEE802.11a/n).

Specifics and differences of IEEE802.11ac than IEEE802.11n are:

- 80 MHz channel width minimum with 160 MHz maximum channel width
- Eight MIMO spatial streams:
 - Introduced downlink MU-MIMO with up to four downlink clients
 - Multiple STAs with multiple antennas can now transmit and receive independently on multiple streams
- 256-QAM optional modulation with the ability to use 1024-WAM
- Standardized beamforming ability

Multi-user MIMO is worth further detail. 802.11ac extends 802.11n from four spatial streams to eight. One of the largest contributing factors to 802.11ac speed is **Spatial Division Multiplexing (SDM)**, referred to earlier. When combined with multi-user or multiple client aspects of 802.11ac, this technique goes by the name **Spatial Diversity Multiple Access (SDMA)**. Essentially, MU-MIMO in 802.11ac is a wireless analog of a network switch. The following illustration depicts an 802.11ac 4 x4: 4 MU-MIMO system with 3 clients.



802.11ac MU-MIMO Usage

802.11.ac also extends the modulation constellation from 64-QAM to 256-WAM. This implies there are 16 amplitude levels and 16 phase angles, requiring very precise hardware to implement. 802.11n represented six bits per symbol, while 802.11ac represents a full eight bits per symbol.

Beamforming methods have been formally standardized by the IEEE committee. For example, the committee has agreed that explicit feedback is the standard approach to beamforming association. This will allow beamforming and the performance benefits to be available from multiple vendors.

The increased bandwidth per channel (up to 80 MHz with the option of 160 MHz or two 80 MHz blocks) provides a substantial increase in throughput in the 5 GHz space.

Theoretically, using an 8 x 8: 8 device, 160 MHz wide channels, and 256-QAM modulation, one could sustain 6.933 GBps throughput in aggregate.

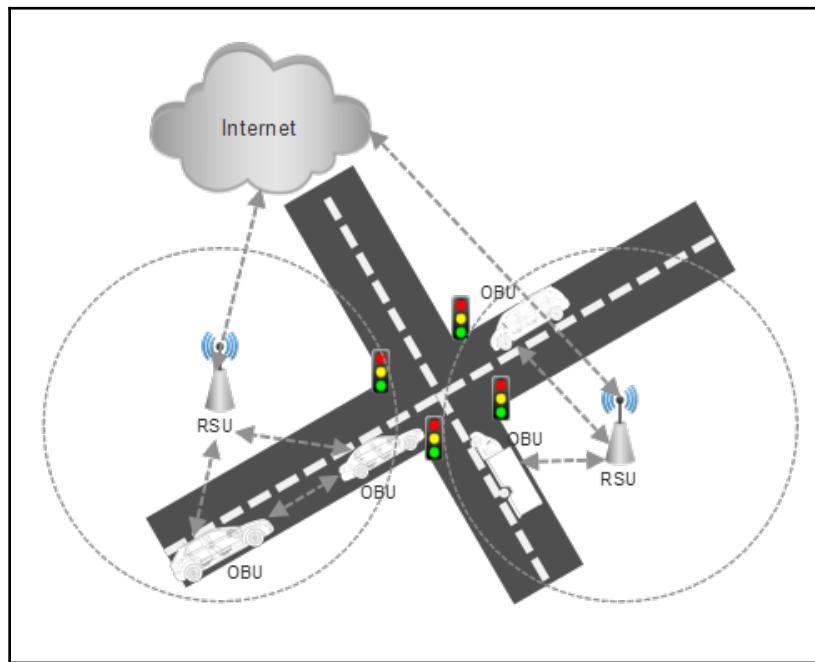
IEEE 802.11p vehicle-to-vehicle

Vehicular networks (sometimes called vehicular ad hoc networks or VANET) are spontaneous and unstructured, operating as a car moves around a city while interacting with other vehicles and infrastructure. This model of a network makes use of **vehicle-to-vehicle (V2V)** and **vehicle-to-infrastructure (V2I)** models.

In 2004, the 802.11p task group formed and developed the first draft by April 2010. 802.11p is considered a **Dedicated Short Range Communication (DSRC)** within the US Department of Transportation. The goal of this network is to provide a standard and secure V2V and V2I system used for vehicular safety, toll collection, traffic status/warnings, roadside assistance, and e-commerce within a vehicle.

The topology and general use case for an IEEE 802.11p network is shown in the following figure. There are two types of nodes in the network. First is the **road-side unit (RSU)**, which is a fixed location device much like an access point. It services to bridge vehicles and moving devices to the internet for application service usage and access to trust authorities. The other node type is the **on-board unit (OBU)**, which resides in the vehicle. It is capable of communicating to other OBUs and the fixed RSUs when needed.

OBUs can communicate with RSUs and each other to relay vehicle and safety data. RSUs are used to bridge to application services and trust authorities for authentication. The following is an example 802.11p usage and topology:



IEEE 802.11p Use Case. Shown are OBU within vehicles and fixed infrastructure RSU.

For transportation systems, there exist several challenges in wireless communication. There is a heightened level of safety necessary to be used in vehicular communication and control. Physical effects such as Doppler shifts, latency effects, and robust ad-hoc networking are a few issues to consider.

Many of the differences from the 802.11 standards are to ensure quality and range over the speed of transmission. Other factors are changes to reduce the latency to start exchanging data. The following is a summary of the features of IEEE 802.11p and differences with the IEEE 802.11a standards:

- Channel width is 10 MHz instead of 20 MHz used in 802.11a.
- IEEE 802.11p operates in the 75 MHz bandwidth of the 5.9 GHz space. This implies there are a total of seven channels available (one control channel, two critical, and four services channels).
- It supports half of the bit rates as compared to 802.11a; that is, 3/4.5/6/9/12/18/24/27 Mbps.

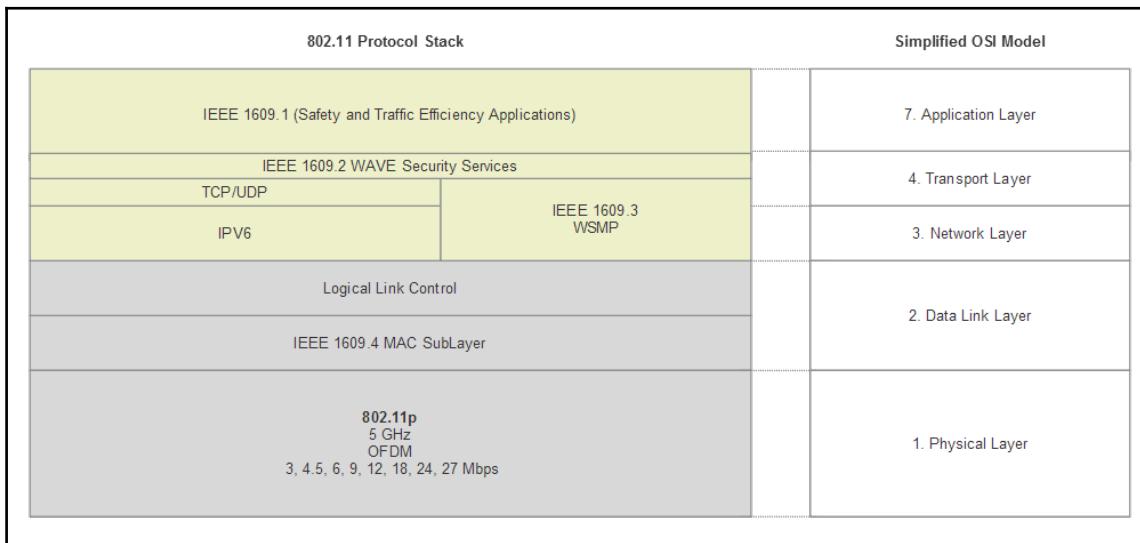
- It has the same modulation schemes such as BPSK/QPSK/16QAM/64QAM and 52 subcarriers.
- The symbol duration has become double compared to 802.11a: IEEE 802.11p supports 8 μ s while 11a supports 4 μ s.
- Guard time interval is 1.6 μ s in 802.11p while 11a has 0.8 μ s.
- Techniques such as MIMO and beamforming are not necessary or part of the specification.

The 75 MHz channels are categorized as follows:

Channel	172	174	176	178	180	182	184
Center frequency (GHz)	5.860	5.870	5.880	5.890	5.900	5.910	5.920
Purpose	Critical life safety	Service	Service	Control	Service	Service	High power public safety

The fundamental usage model of 802.11p is to create and associate to ad hoc networks rapidly. This link will come and go as vehicles move at highway speeds in opposite directions. In the standard 802.11 model, a BSS would be the network topology to use, which requires the synchronization, association, and authentication for a wireless network to form. 802.11p provides a wildcard BSSID in the header of all frames exchanged, and are allowed to start exchanging data frames immediately after arriving on a communication channel.

The IEEE 802.11p protocol stack is derived from 802.11a, but makes important changes to address vehicular safety and security. The following figure depicts the protocol stack. A significant departure from other IEEE 802.11 stacks is the use of the IEEE 1609.x standards to address application and security models. The full stack is called the **Wireless Access in Vehicular Environments (WAVE)** and combines 802.11p PHY and MAC layers with IEEE 1609.x layers:



Specific differences to highlight in the stack include:

- **1609.1:** Wave resource manager. Allocates and provisions resources as needed.
- **1609.2:** Defines security services for application and management messages. This layer also provides two modes of encryption. A public-key algorithm using the signing algorithm ECDSA can be used. Alternatively, a symmetric algorithm based on AES-128 in CCM mode is available.
- **1609.3:** Assists with connection setup and management of WAVE compliant devices.
- **1609.4:** Provides multi-channel operation above the 802.11p MAC layer.

Security is critical in a VANET. There are potential threats that can directly affect public safety. Attacks can occur when bogus information is broadcast affecting how other vehicles react or perform. An attack of this nature could be a rogue device broadcasting a hazard in a road causing vehicles to hard stop. Security also needs to account for private vehicular data, masquerading as other vehicles and causing a denial of services attacks. All these can lead to catastrophic events and need standards such as the IEEE 1609.2.

IEEE 802.11ah

Based on 802.11ac architecture and PHY, 802.11ah is a variant of the wireless protocols targeted for the IoT. The design attempts to optimize for constrained sensor devices that require long battery life and can optimize range and bandwidth. 802.11ah is also referred to as HaLow, which is essentially a play on words with "ha" referring to "ah" backward and low implying low power and lower frequency. Put together, it forms a derivative of "hello".

The intent of the IEEE 802.11ah task group was to create a protocol with extended range for rural communications and offloading cell traffic. The secondary purpose was to use the protocol for low throughput wireless communications in the sub-gigahertz range. The specification was published on December 31, 2016. The architecture is most different from other forms of 802.11 standards, in the following ways, in particular:

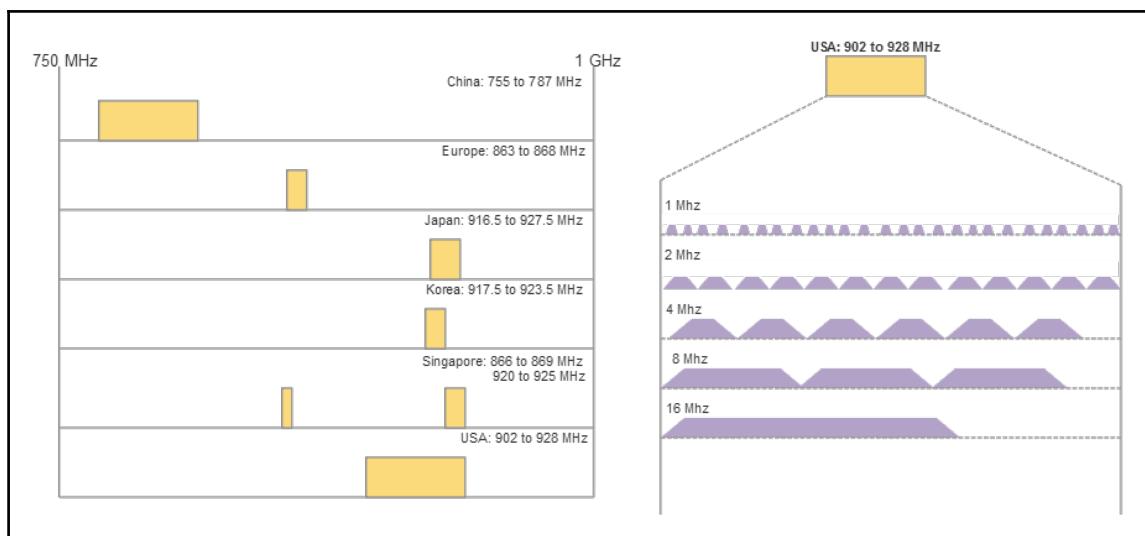
- Operates in 900 MHz spectrum. This allows for good propagation and penetration of materials and atmospheric conditions.
- Channel width varies and can be set to 2, 4, 8, or 16 MHz-wide channels.
- The available modulation methods are diverse and include BPSK, QPSK, 16-QAM, 64-QAM, and 256-QAM modulation techniques.
- Modulation based on 802.11ac standard with specific changes. A total of 56 OFDM subcarriers with 52 dedicated to data and four dedicated to pilot tones.
- Total symbol duration is 36 or 40 microseconds.
- Supports SU-MIMO beamforming.
- Fast association for networks of thousands of STAs using two different authentication methods to limit contention.
- Provides connectivity to thousands of devices under a single access point.
- Includes the ability to relay to reduce power on STAs and allow for a crude form of mesh networking using a one-hop reach method.
- Allows for advanced power management on each 802.11ah node.
- Allows for non-star topology communication through the use of Restricted Access Windows.
- Allows for sectorization, which enables antennas to be grouped to cover different regions of a BSS (called sectors). This is accomplished using beamforming adopted from other 802.11 protocols.

The minimum throughput will be 150 kbps, based on BPSK modulation on a single MIMO stream at 1 MHz channel bandwidth. The maximum theoretical throughput will be 347 Mbps based on a 256-WAM modulation using 4 MIMO streams and 16 MHz channels.



The IEEE 802.11ah specification requires that STAs support 1MHz and 2 GHz channel bandwidths. The access points must support 1, 2, and 4 MHz channels. 8Mhz and 16 Mhz channels are optional. The narrower the channel bandwidth, the longer the range but the slower the throughput. The wider the channel bandwidth, the shorter the range but the faster the throughput.

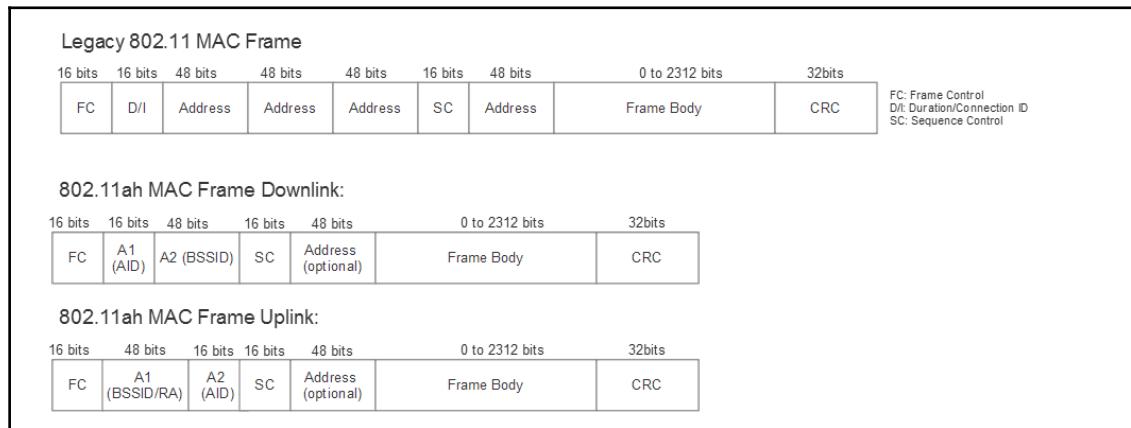
Channel width will vary depending on the region in which 802.11ah is deployed. Some combinations will not work due to regulations in specific regions as shown:



Left: Different channelization options depending on regional regulations. Right: varying bandwidth options and channel bonding within the US region from 1MHz to 16 MHz channels.

Every attempt in the architecture of the IEEE 802.11ah standard is aimed at optimizing overall range and efficiency. This reaches down as far as the length of the MAC headers.

The goal to connect several thousand devices to a single AP is also accomplished using a unique association identifier (AID) assignment of 13 bits. This allows for grouping of STAs based on criteria (hallway lights, light switch, and so on). This allows an AP to connect to over 8191 STAs (802.11 could support only 2007 STAs). That many nodes, however, has the potential to induce a massive number of channel collisions. Even though the number of connected STAs increased, the goal was to reduce the amount of data in transit to address these stations. The IEEE task group accomplished this by removing a number of fields that were not particularly relevant for the IoT use cases, such as the QoS and DS fields. The following figure illustrates the 802.11ah MAC downlink and uplink frames as compared to standard 802.11.



A comparison of the standard 802.11 MAC frame and 802.11ah condensed frames.

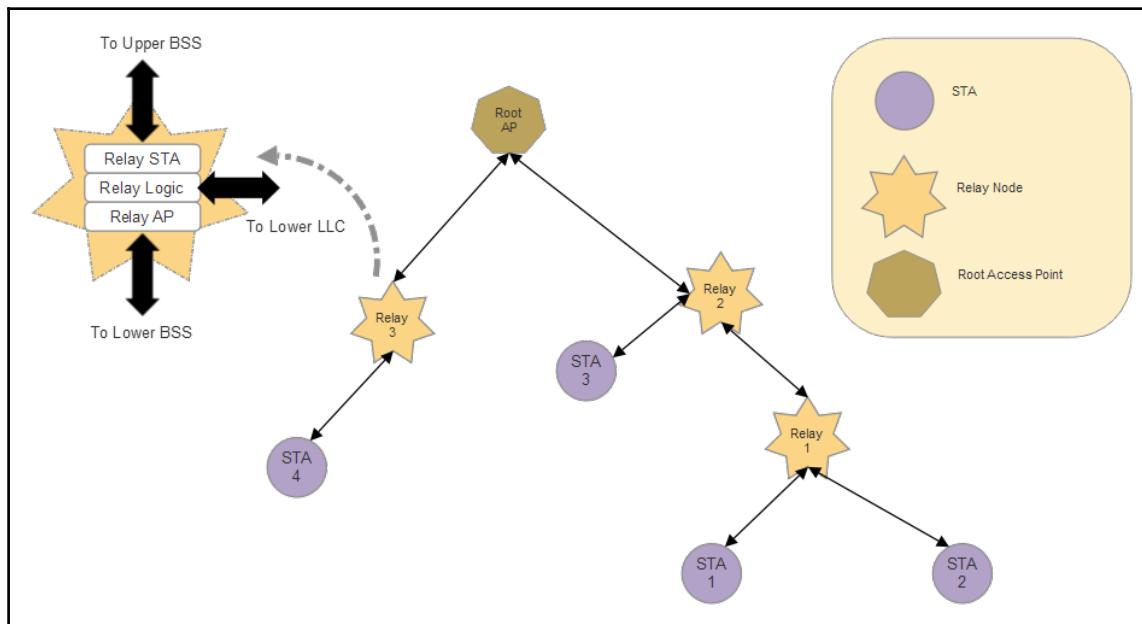
Another improvement for power management and channel efficiency is attributed to removing acknowledgment frames. ACKs are implicit for bidirectional data. That is if both devices are sending and receiving data from each other. Normally an ACK would be used after the successful reception of a packet. In this bidi (BDT) mode, reception of the next frame implies that the previous data was successfully received and no ACK packet needs to be exchanged.

To avoid a sea of collisions which would prevent a functional network, 802.11ah uses a **Restricted Access Window (RAW)**. As the STAs are divided into various groups using the AID, channels will be split into time slots. Each group would be assigned a specific time slot and no others. There are exceptions, but for the general case, the grouping forms an arbitrary isolation. The additional benefit of RAW is that devices can enter a hibernation state to conserve power whenever it isn't their time slot for transmission.

Topology-wise, there are three types of stations in an 802.11ah network:

- **Root access point:** The principal root. Typically, serves as a gateway to other networks (WAN).
- **STA:** The typical 802.11 station or end point client.
- **Relay node:** A special node that combines an AP interface to STAs residing to a lower BSS and an STA interface to other relay nodes or a root AP on the upper BSS.

The following figure is the IEEE802.11ah topology. This architecture differs substantially from other 802.11 protocol in the use of a single hop relay nodes that act to create identifiable BSS. The hierarchy of relays forms a larger network. Each relay acts as an AP and STA.



IEEE802.11ah Network Topology.

In addition to the basic node types, there are three power saving states an STA can reside in:

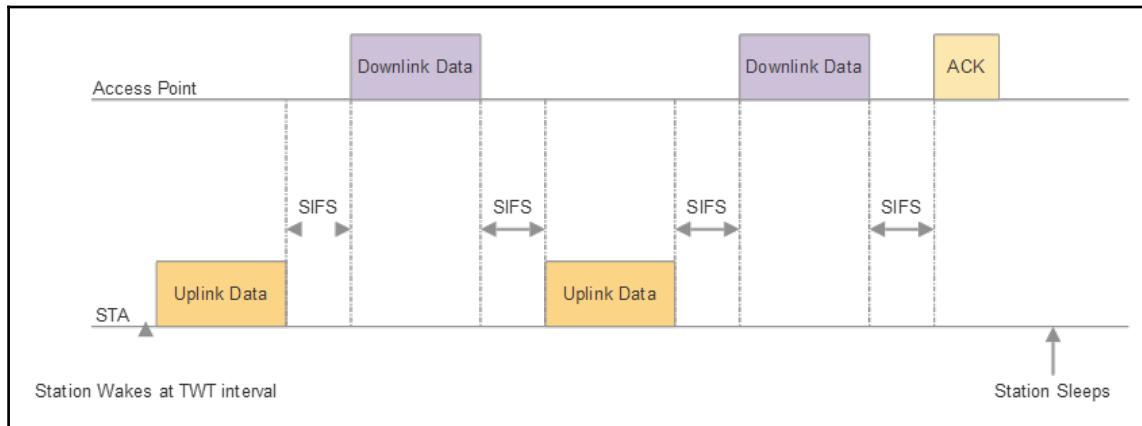
- **Traffic Indication Map (TIM):** Listens to AP for data transfer. Nodes will periodically receive information about data buffered for them from its access point. The message sent is called the TIM Information Element.
- **Non-TIM stations:** Negotiates with AP directly during association to obtain transmission time on **Periodic Restricted Access Windows (PRAW)**.
- **Unscheduled stations:** Does not listen to any beacons and uses polling to access channels.

Power is critical in IoT sensor and edge devices based on coin cell batteries or energy harvesting. 802.11 protocols are notorious for high power demands. To remediate the power of this wireless protocol, 802.11ah uses a Max Idle Period value, which is part of the regular 802.11 specifications. In a general 802.11 network, the Max Idle Period is roughly 16 hours based on a time of 16-bit resolution. In 802.11ah, the first two bits of the 16-bit timer are scaling factor that allows the sleep duration to exceed five years.

Additional power is mitigated through changes to the beacon. As previously covered, beacons relay information on the availability of buffered frames. Beacons will carry a TIM bitmap, which inflates their size since 8191 STAs will cause the bitmap to grow substantially. 802.11ah uses a concept called TIM segmentation where some beacons carry portions of the overall bitmap. Each STA calculates when their respective beacon with bitmap information will arrive and allows the device to enter a power save mode until the moment it needs to wake and receive beacon information.

Another power-save feature is called the **Target Wake Time (TWT)** and is intended for STAs that rarely transmit or receive data. This is very common in IoT deployments such as temperature sensor data. An STA and its associated AP will negotiate to arrive at an agreed upon TWT and the STA will enter a sleep state until that timer is signaled.0;

The process of implicit ACKs is called a *Speed Frame Exchange*, and it is shown in the following figure:



IEEE 802.11ah Speed Frame Exchange: Example of Target Wake Time (TWT) used to start an STA communicating. SIFS represents the gap between the AP and STA communicating. No ACKs are used between data pairs. Only a single ACK at the end of transmission is sent before the STA returns to a sleep mode.

Summary

This chapter covered a necessary portion of IoT communication. Using IP-based standard communication greatly simplifies design and allows for rapid and easy scaling. Scaling is critical for IoT deployments that reach into thousands or millions of nodes. Using an IP-based transport allows for common tools to simply just work. 6LoWPAN and Thread demonstrate standards that can be applied to traditionally non-IP protocols such as 802.15.4. Both protocols allow for IPv6 addressing and mesh networking to massive IoT networks. 802.11 is a significant and extremely successful protocol that forms the basis of the WLAN, but can also reach into IoT devices and sensors using 802.11ah or transportation systems using 802.11p. The following table contrasts a non-IP traditional protocol to an IP protocol. Typically, the difference will be in power, speed, and range.

The architect needs to balance these parameters to deploy the correct solution:

	802.15.4	802.11ah
IP Base	Non-IP based (need 6LoWPAN or Thread)	IP-based
Range	100 m	Targeted to 1000 m
Network structure	Full mesh	Hierarchical with a single node hop
Channelization	ISM 2.4 GHz with only DSSS	Sub 1 GHz ISM with various modulation coding schemes. Channel bandwidth: 1,2, 4, 8, 16 MHz
Channel interference management	CSMA/CA	RAW mechanism allowing STAs to associate group based time slots
Throughput	250 Kbps	150 kbps to 347 Mbps
Latency	Good	Best (2x better than 802.15.4)
Energy efficiency	Best (17 mJ/Packet)	Good (63 mJ/packet)
Power savings	Sleep-wake mechanisms in frames.	Multiple data structures to control and fine-tune power at various levels
Network size	Possible to 65,000	8192 STAs

The next chapter will go into the very far range protocols or the wide area network. This will include traditional cellular (4G LTE) and IoT cellular models such as Cat1. The chapter will also discuss LPWAN protocols such as Sigfox and LoRa. The WAN is the next necessary component for getting data onto the internet.

7

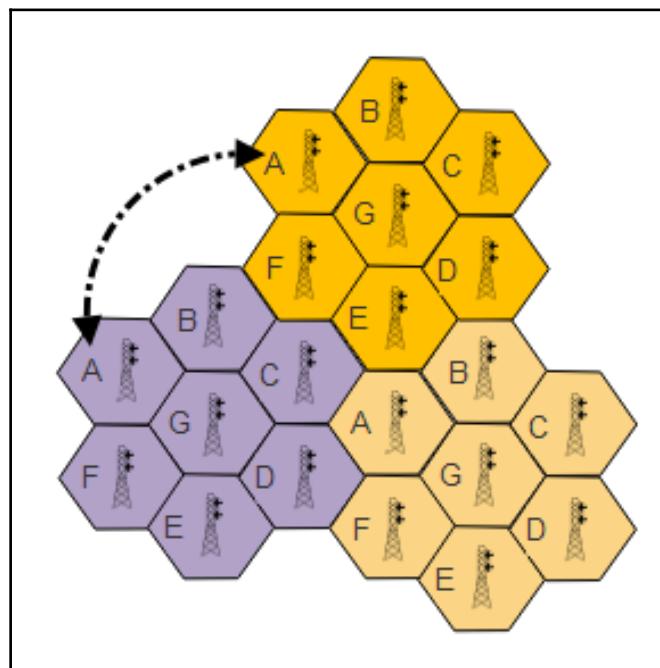
Long-Range Communication Systems and Protocols (WAN)

So far, we have discussed **Wireless Personal Area Networks (WPAN)** and **Wireless Local Area Networks (WLAN)**. These types of communication bridge the sensors to a local net but not necessarily the internet or other systems. We need to remember that the IoT ecosphere will include sensors, actuators, cameras, smart-embedded devices, vehicles, and robots in the remotest of places. For the long haul, we need to address the **Wide Area Network (WAN)**.

This chapter covers the various WAN devices and topologies including cellular (4G-LTE and the upcoming 5G standard) as well as other proprietary systems including **Long Range Radio (LoRa)** and Sigfox. While this chapter will cover cellular and long-range communication systems from a data perspective, it will not focus on the analog and voice portions of mobile devices. Long-range communication is usually a service, meaning it has a subscription to a carrier providing cellular tower and infrastructure improvements. This is different to the previous WPAN and WLAN architectures as they are usually contained in a device that the customer or developer produces or resells. A subscription or service-level agreement (SLA) has another effect on the architecture and constraints of systems that need to be understood by the architect.

Cellular connectivity

The most prevalent communication form is cellular radio and specifically cellular data. While mobile communication devices, had existed for many years before cellular technology, they had limited coverage, shared frequency space, and were essentially two-way radios. Bell Labs built some trial mobile phone technologies in the 1940s (Mobile Telephone Service) and 1950s (Improved Mobile Telephone Service) but had very limited success. There were also no uniform standards for mobile telephony at the time. It wasn't until the cellular concept was devised by Douglas H. Ring and Rae Young in 1947 and then built by Richard H. Frenkiel, Joel S. Engel, and Philip T. Porter at Bell Labs in the 1960s that larger and robust mobile deployments could be realized. The handoff between cells was conceived and built by Amos E. Joel Jr. also of Bell Labs, which allowed for handoff when moving cellular devices. All these technologies combined to form the first cellular telephone system, first cellular phone, and the first cellular call made by Martin Cooper of Motorola on April 3, 1979. Following is an ideal cellular model where cells are represented as hexagonal areas of optimal placement.



Cellular Theory. The hexagonal pattern guarantees separation of frequencies from the nearest neighbors. No two similar frequencies are within one hex space from each other, as shown in the case of frequency A in two different regions. This allows for frequency reuse.

The technologies and proof of concept designs eventually led to the first commercial deployments and public acceptance of mobile telephone systems in 1979 by NTT in Japan, and then in Denmark, Finland, Norway, and Sweden in 1981. The Americas didn't have a cell system until 1983. These first technologies are known as 1G, or the first generation of cellular technology. A primer for the generations and their features will be detailed next, however, the next section will specifically describe 4G-LTE as that is the modern standard for cellular communication and data. The following sections will describe other IoT and future cellular standards such as NB-IOT and 5G.

Governance models and standards

The **International Telecommunication Union (ITU)** is a UN specialized agency and was founded in 1865; it took its present name in 1932, before becoming a specialized agency in the UN. It plays a significant role worldwide in wireless communication standards, navigation, mobile, internet, data, voice, and next-gen networks. It includes 193 member nations and 700 public and private organizations. It too has a number of working groups called sectors. The sector relevant to cellular standards is the **Radiocommunication Sector (ITU-R)**. The ITU-R is the body that defines the international standards and goals for various generations of radio and cellular communication. These include reliability goals and minimum data rates.



The ITU-R has produced two fundamental specifications that have governed cellular communication in the last decade. The first was the **International Mobile Telecommunications-2000 (IMT-2000)**, which specifies the requirements for a device to be marketed as 3G. More recently, the ITU-R produced a requirement specification called **International Mobile Telecommunications-Advanced (IMT-Advanced)**. The IMT-Advanced system is based on an all-IP mobile broadband wireless system. The IMT-Advanced defines what can be marketed as 4G worldwide. The ITU was the group that approved of **Long-Term Evolution (LTE)** technology in the 3GPP roadmap to support the goals of 4G cellular communication in October of 2010. The ITU-R continues to drive the new requirements for 5G.

Examples of the ITU-Advanced set of requirements for a cellular system to be labeled 4G include:

- Must be an all-IP, packet-switched network
- Interoperable with existing wireless

- A nominal data rate of 100 Mbps when the client is moving and 1 Gbps while the client is fixed
- Dynamically share and use network resources to support more than one user per cell
- Scalable channel bandwidth of 5 to 20 MHz
- Seamless connectivity and global roaming across multiple networks

The issue is that often the entire set of ITU goals are not met, and there exists naming and branding confusion:

Feature	1G	2/2.5G	3G	4G	5G
First Availability	1979	1999	2002	2010	2020
ITU-R Specification	NA	NA	IMT-2000	IMT-Advanced	IMT-2020
ITU-R Frequency Specification	NA	NA	400 MHZ to 3 GHz	450 MHz to 3.6 GHz	TBD
ITU-R Bandwidth Specification	NA	NA	Stationary: 2 Mbps Moving: 384 Kbps	Stationary: 1 Gbps Moving: 100 Mbps	Min Down: 20 Gbps Min Up: 10 Gbps
Typical Bandwidth	2 Kbps	14.4-64 Kbps	500 to 700 Kbps	100 to 300 Mbps (peak)	TBD?
Usage/Features	Mobile telephony only.	Digital voice, SMS text, caller-ID, one-way data.	Superior audio, video, and data. Enhanced roaming.	Unified IP and seamless LAN/WAN/WLAN.	IoT, ultra density, low latency.

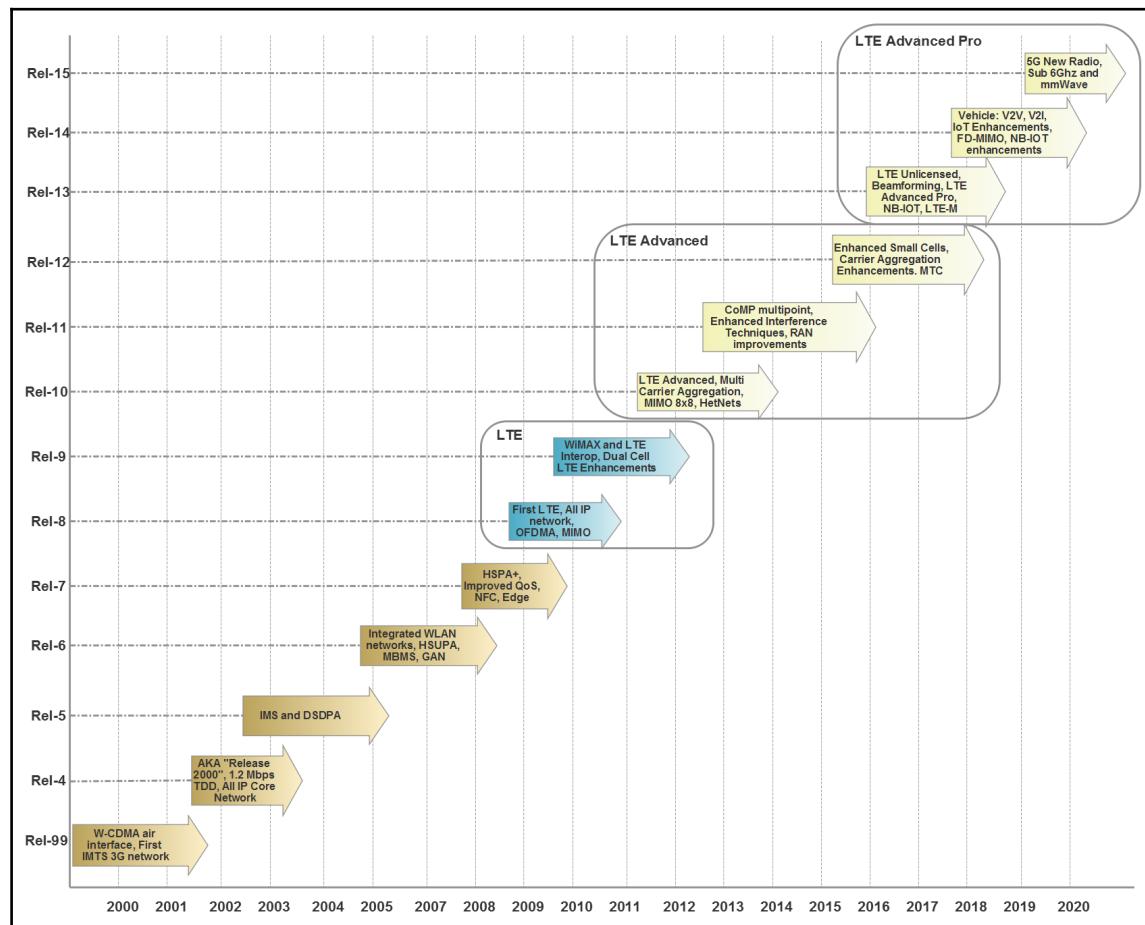
Standards and Multiplexing	AMPS	2G: TDMA, CDMA, GSM 2.5G: GPRS, EDGE, 1xRTT	FDMA, TDMA WCDMA, CDMA-2000	CDMA	CDMA
Handoff	Horizontal	Horizontal	Horizontal	Horizontal and vertical	Horizontal and vertical
Core Network	PSTN	PSTN	Packet Switch	Internet	Internet
Switching	Circuit	Circuit for access network and air network	Packet-based except for air interface	Packet-based	Packet-based
Technology	Analog Cellular	Digital Cellular	Broad bandwidth CDMA, WiMAX, IP-based	LTE Advanced Pro-based	LTE Advanced Pro-based, mmWave

The other standard body in the cellular world, 3GPP, is the acronym for the Third Generation Partnership Project and is the group of seven telecom organizations (also known as the Organizational Partners) from across the globe that manage and govern cellular technology. The group formed in 1998 with the partnership of Nortel Networks and AT&T Wireless and released the first standard in 2000. Organizational Partners and Market Representatives contribute to 3GPP from Japan, the USA, China, Europe, India, and Korea. The overall goal of the group is to recognize standards and specifications for the **Global System for Mobile Communications (GSM)** in the creation of the 3G specifications for cellular communication. 3GPP work is performed by three **Technical Specification Groups (TSG)** and six **Working Groups (WG)**. The groups meet several times per year in different regions. The main focus of 3GPP releases is to make the system backward and forward compatible (as much as possible).



There is a degree of confusion in the industry as to the differences between the ITU, 3GPP, and LTE definitions. The easiest way to conceptualize the relationship is that the ITU will define the goals and standards worldwide for a device to be labeled 4G or 5G. The 3GPP responds to goals with technologies such as the family of LTE improvements. The ITU still ratifies that such LTE advancements meet their requirements to be labeled 4G or 5G.

The figure below shows the 3GPP technology releases since 2000. Boxed are the LTE evolution technologies.



3GPP Releases from 2000 to 2020.



LTE and its role in the cellular vernacular are also often confused. LTE stands for Long-Term Evolution and is the path followed to achieve ITU-R speeds and requirements (which are initially quite aggressive). Mobile phone vendors will release new smartphones in an existing cellular neighborhood using legacy backend technology such as 3G. Carriers advertise 4G-LTE connectivity if they demonstrated a substantial improvement in speed and features over their legacy 3G networks. In the mid to late-2000s, many carriers did not meet the ITU-R 4G specification but essentially got close enough. Carriers used legacy technologies and rebranded themselves as 4G in many cases. LTE-Advanced is yet another improvement that gets even closer to ITU-R goals.

In summary, the terminology can be confusing and misleading and an architect needs to read beyond the branding labels to understand the technology.

Cellular access technologies

It is important to understand how cellular systems work with multiple users of voice and data. There are several standards worth reviewing, similar to concepts covered for WPAN and WLAN systems. Before LTE was supported by the 3GPP, there were multiple standards for cellular technology, particularly GSM devices and CDMA devices. It should be noted that these technologies are incompatible with each other, from infrastructure to the device:

- **Frequency Division Multiple Access (FDMA):** Common in analog system systems but rarely used today in digital domains. It is a technique whereby the spectrum is divided up into frequencies and then assigned to users. One transceiver at any given time is assigned to a channel. The channel, therefore, is closed to other conversations until the initial call is finished, or until it is handed off to a different channel. A “full-duplex” FDMA transmission requires two channels, one for transmitting and one for receiving.

- **Code Division Multiple Access (CDMA):** Based on spread spectrum technology. CDMA increases spectrum capacity by allowing all users to occupy all channels at the same time. Transmissions are spread over the whole radio band, and each voice or data call is assigned a unique code to differentiate from the other calls carried over the same spectrum. CDMA allows for a soft hand-off, which means that terminals can communicate with several base stations at the same time. The dominant radio interface for third-generation mobile was originally designed by Qualcomm as CDMA2000 and targeted 3G. Because of its proprietary nature, it didn't achieve global adoption and is used in less than 18% of the global market. It manifested in the US with Verizon and Sprint being strong CDMA carriers.
- **Time Division Multiple Access (TDMA):** In this model, the capacity of the spectrum increases by splitting each frequency into time slots. TDMA allows each user to access the entire radio frequency channel for the short period of a call. Other users share this same frequency channel at different time slots. The base station continually switches from user to user on the channel. TDMA is the dominant technology for second-generation mobile cellular networks. The GSM organization adopted TDMA as a multi-access model. It resides in four distinct bands: 900 MHz/1800 MHz in Europe and Asia and 850 MHz/1900 MHz in North and South America.

Some devices and modems may still support GSM/LTE or CDMA/LTE. GSM and CDMA are incompatible with each other. GSM/LTE and CDMA/LTE can be compatible, however, if they support the LTE bands. In older devices, voice information is delivered on a 2G or 3G spectrum, which are quite different for CDMA and GSM (TDMA). Data, too, is incompatible since LTE data runs on 4G bands.

3GPP user equipment categories

In release 8, there were five types of user equipment categories, each with different data rates and MIMO architectures. Categories allowed 3GPP to differentiate between LTE evolutions. Since release 8, more categories have been added. Categories combine the uplink and downlink capabilities as specified by the 3GPP organization. Typically, you will acquire a cellular radio or chipset that labels the category it is capable of supporting. While user equipment may support a particular category, the cell system (eNodeB, discussed later) must also support the category.

Part of the association process between a cell device and infrastructure is the exchange of capability information such as the category:

3GPP Release	Category	Max L1 Downlink Datarate (Mbps)	Max L1 Uplink Datarate (Mbps)	Max Number of Downlink MIMO
8	5	299.6	75.4	4
8	4	150.8	51	2
8	3	102	51	2
8	2	51	25.5	2
8	1	10.3	5.2	1
10	8	2,998.60	1,497.80	8
10	7	301.5	102	2 or 4
10	6	301.5	51	2 or 4
11	9	452.2	51	2 or 4
11	12	603	102	2 or 4
11	11	603	51	2 or 4
11	10	452.2	102	2 or 4
12	16	979	n/a	2 or 4
12	15	750	226	2 or 4
12	14	3,917	9,585	8
12	13	391.7	150.8	2 or 4
12	0	1	1	1
13	NB1	0.68	1	1
13	M1	1	1	1
13	19	1566	n/a	2, 4, or 8
13	18	1174	n/a	2, 4, or 8
13	17	25,065	n/a	8



Notice Cat M1 and Cat NB1 in release 13. Here the 3GPP organization reduced the data rates significantly to 1 Mbps or lower. These are the categories dedicated to IoT devices that need low data rates and only communicate in short bursts.

4G-LTE spectrum allocation and bands

There are 55 LTE bands in existence, partly due to spectrum fragmentation and market strategies. The proliferation of LTE bands is also a manifestation of government allocation and the auctioning of frequency space. LTE is also split into two categories that are not compatible:

- **Time Division Duplex (TDD):** TDD uses a single frequency space for uplink and downlink data. The transmission direction is controlled via time slots.
- **Frequency Division Duplex (FDD):** In an FDD configuration, the base station (eNodeB) and the **user equipment (UE)** will open a pair of frequency spaces for uplink and downlink data. An example may be LTE band-13, which has an uplink range of 777 to 787 MHz and a downlink range of 746 to 756 MHz. Data can be sent simultaneously to both the uplink and downlink.

Combination TDD/FDD modules exist that combine both technologies into a single modem allowing for multiple carrier usages.

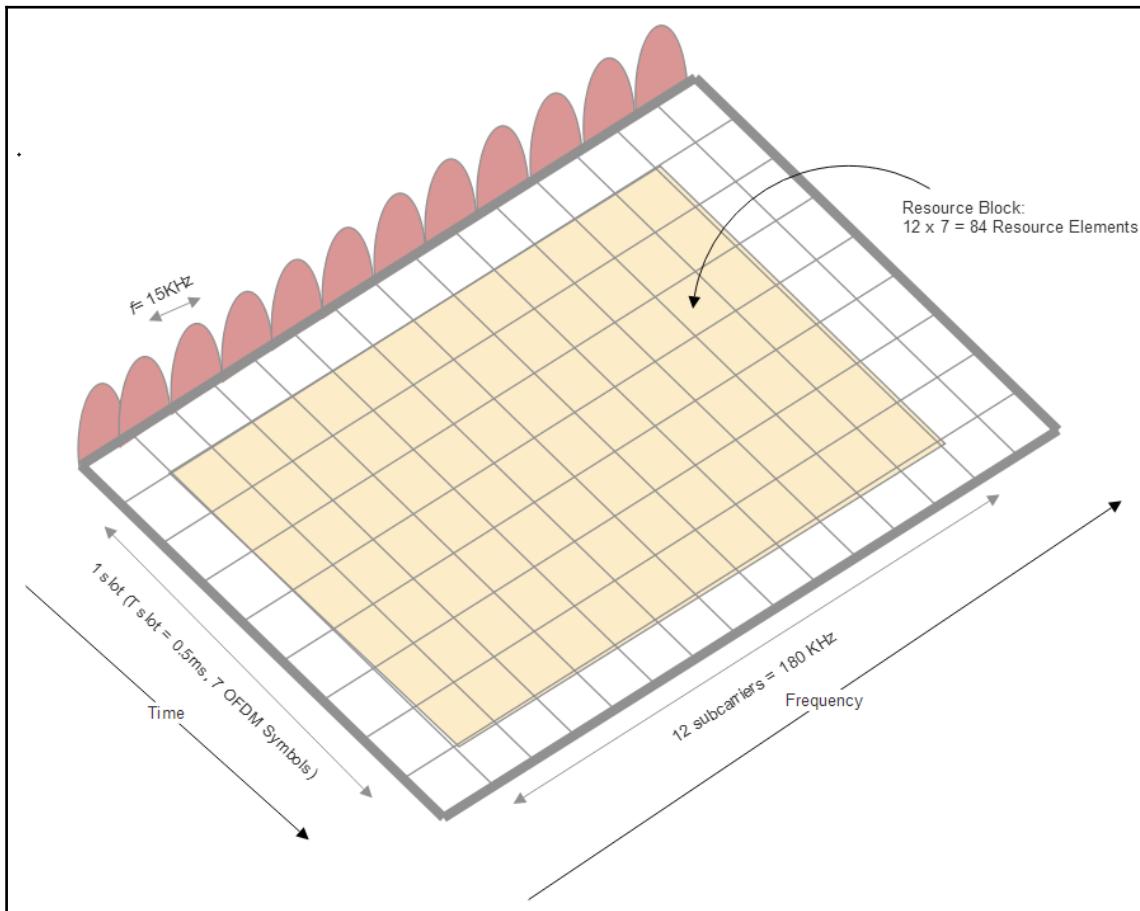


Note that the downlink will always refer to communication from the eNodeB to the user equipment, and uplink will refer to the opposite direction.

A few other terms specific to LTE are needed to understand the spectrum usage:

- **Resource Element:** This is the smallest transmission unit in LTE. The RE consists of one subcarrier for exactly one unit of symbol time (OFDM or SC-FDM).
- **Subcarrier spacing:** This is the space between subcarriers. LTE uses 15 KHz spacing with no guard bands.
- **Cyclic prefix:** Since there are no guard bands, a cyclic prefix time is used to prevent multipath inter-symbol interference between subcarriers.
- **Time slot:** LTE uses a 0.5 ms time period for LTE frames. That equals six or seven OFDM symbols depending on the cyclic prefix timing.
- **Resource block:** One unit of transmission. Contains 12 subcarriers and seven symbols, which is equal to 84 resource elements.

An LTE frame which is 10 ms long would consist of 10 subframes. If 10% of the total bandwidth of a 20 MHz channel is used for a cyclic prefix, then the effective bandwidth is reduced to 18 MHz. The number of subcarriers in 18 MHz is $18 \text{ MHz}/15 \text{ kHz} = 1200$. The number of resource blocks is $18 \text{ MHz}/180 \text{ kHz} = 100$:



One slot of an LTE frame. A 10ms LTE frame consists of 20 slots. Slots are based on twelve 15 KHz spaced subcarriers and seven OFDM symbols. This combines into $12 \times 7 = 84$ resource elements.

The bands allocated for 4G-LTE are specific to regional regulations (North America, Asia-Pacific, and so on). Each band has a set of standards developed and ratified by the 3GPP and ITU. Bands are split between FDD and TDD areas and have common name acronyms routinely used in the industry, such as **Advanced Wireless Service (AWS)**. The following is a table separating the FDD and TDD bands for the North American region only:

Band	Duplex	f (MHz)	Common Name	North America	Width of Band (MHz)	Duplex Space (MHz)	Band Gap (MHz)
1	FDD	2100	IMT		60	190	130
2	FDD	1900	PCS blocks A-F	Yes	60	80	20
3	FDD	1800	DCS		75	95	20
4	FDD	1700	AWS blocks A-F (AWS-1)	Yes	45	400	355
5	FDD	850	CLR	USA (AT&T, U.S. Cellular)	25	45	20
6	FDD				10	35	25
7	FDD	2600	IMT-E	Canada (Bell, Rogers, Telus)	70	120	50
8	FDD	900	E-GSM		35	45	10
9	FDD				35	95	60
10	FDD	1700	Extended AWS blocks A-I		60	400	340
11	FDD	1500	Lower PDC	Canada (Bell), Guam (iConnect, ...)	20	48	28
12	FDD	700	Lower SMH blocks A/B/C	USA (Verizon), Canada (Bell, EastLink, ...)	18	30	12
13	FDD	700	Upper SMH block C	USA (FirstNet)	10	-31	41
14	FDD	700	Upper SMH block D		10	-30	40
15	FDD	2000			20	700	680
16	FDD	700			15	575	560
17	FDD	700	Lower SMH blocks B/C	Canada (Rogers), Guam (NTT), USA (AT&T)	12	30	18
18	FDD	850	Japan lower 800		15	45	30
19	FDD	850	Japan upper 800		15	45	30
20	FDD	800	EU Digital Dividend		30	-41	71
21	FDD	1500	Upper PDC		15	48	33
22	FDD	3500		USA (Ligado Networks)	90	100	10
23	FDD	2000			20	180	160
24	FDD	1600	L-Band (US)		34	-101.5	135.5
25	FDD	1900	Extended PCS blocks A-G	USA (Sprint)	65	80	15
26	FDD	850	Extended CLR	USA (Sprint)	30 / 40		10
27	FDD	800	SMR		17	45	28
28	FDD	700	APT		45	55	10
29	FDD[A 1]	700	Lower SMH blocks D/E	USA (AT&T)	11	n/a	
30	FDD	2300	WCS blocks A/B	USA (AT&T)	10	45	35
31	FDD	450			5	10	5
32	FDD[A 1]	1500	L-Band (EU)		44	n/a	
65	FDD	2100	Extended IMT		90	190	
66	FDD	1700	Extended AWS blocks A-J	Canada (Freedom Mobile)	90/70	400	
67	FDD[A 1]	700	EU 700		20	n/a	
68	FDD	700	ME 700		30	55	
69	FDD[A 1]	2600	IMT-E (Duplex spacing)		50	n/a	
70	FDD	2000	AWS-4	USA (DISH)	25/15	300	
71	FDD	600	US Digital Dividend	USA (T-Mobile)			

4G-LTE Frequency Division Duplex band allocation and North American carrier ownership.

Band	Duplex	f (MHz)	Common Name	North America	Allocation (MHz)	Width of Band (MHz)
33	TDD	2100	IMT		1900 – 1920	20
34	TDD	2100	IMT		2010 – 2025	15
35	TDD	1900	PCS (Uplink)		1850 – 1910	60
36	TDD	1900	PCS (Downlink)		1930 – 1990	60
37	TDD	1900	PCS (Duplex spacing)		1910 – 1930	20
38	TDD	2600	IMT-E (Duplex Spacing)		2570 – 2620	50
39	TDD	1900	DCS-IMT gap		1880 – 1920	40
40	TDD	2300			2300 – 2400	100
41	TDD	2500	BRS / EBS	USA (Sprint)	2496 – 2690	194
42	TDD	3500			3400 – 3600	200
43	TDD	3700			3600 – 3800	200
44	TDD	700	APT		703 – 803	100
45	TDD	1500	L-Band (China)		1447 – 1467	20
46	TDD	5200	U-NII		5150 – 5925 (unlicensed)	775
47	TDD	5900	U-NII-4 (V2X)		5855 – 5925 (unlicensed)	70
48	TDD	3600	CBRS		3550 – 3700	150

4G-LTE Time Division Duplex band allocation for North America and North American carrier ownership.

LTE also has been developed to work in the unlicensed spectrum. Originally a Qualcomm proposal, it would operate in the 5 GHz band with IEEE 802.11a. The intent was for it to serve as an alternative to Wi-Fi hotspots. This frequency band of between 5150 MHz and 5350 MHz typically requires radios to operate at 200 mW maximum and indoors only. To this date, only T-Mobile supports the unlicensed use of LTE in areas of the USA. AT&T and Verizon are conducting public tests with LAA mode. There are three categories of unlicensed spectrum use for cellular:

- **LTE Unlicensed (LTE-U):** As mentioned, this would coexist in the 5 GHz band with Wi-Fi devices. The control channel of LTE would remain the same while voice and data would migrate to the 5 GHz band. Within LTE-U is the concept that user equipment may only support unidirectional downlinks in the unlicensed band or full duplex.
- **Licensed-Assisted Access (LAA):** This is similar to LTE-U but governed and designed by the 3GPP organization. It uses a contention protocol called **listen-before-talk (LBT)** to assist in coexisting with Wi-Fi.



A new unlicensed technology called Multifire is also an option in the unlicensed spectrum. Multifire is a form of LTE in the 5GHz space like LTE-U and LAA. Multifire, however, does not need to be anchored in a licensed band. That implies that no commercial carrier (such as AT&T or Verizon) will manage the communication. With the carrier out of the value chain, the economics change for data transport and data ownership since a large company can build and manage their own cellular network using only the unlicensed frequencies.

Multifire uses the same LBT coexistence technology as LAA. In reality, Multifire would need to rely on a blanket of small cells to provide coverage (similar to Wi-Fi access points). Another benefit of Multifire is that the band is globally used by Wi-Fi and allows for consistency between regions.

4G-LTE topology and architecture

The 3GPP LTE architecture is called the **System Architecture Evolution (SEA)**, and its overall goal is to deliver a simplified architecture based on all-IP traffic. It also supports high-speed communication and low latency over **Radio Access Networks (RANs)**. In release 8 of the 3GPP roadmap, LTE was introduced. Since the network is entirely composed of IP packet-switched components, that means that voice data is also sent across as digital IP packets. This is another fundamental difference from the legacy 3G network.



3G topology used circuit switching for voice and SMS traffic and packet switching for data. Circuit switching differs fundamentally from packet switching. Circuit switching manifests from the original telephone switching network. It will use a dedicated channel and path between a source and a destination node for the duration of the communication. In a packet switch network, messages will be broken into smaller fragments (called packets in the case of IP data) and will seek the most efficient route from a source of data to the destination. A header enveloping the packet provides destination information amongst other things.

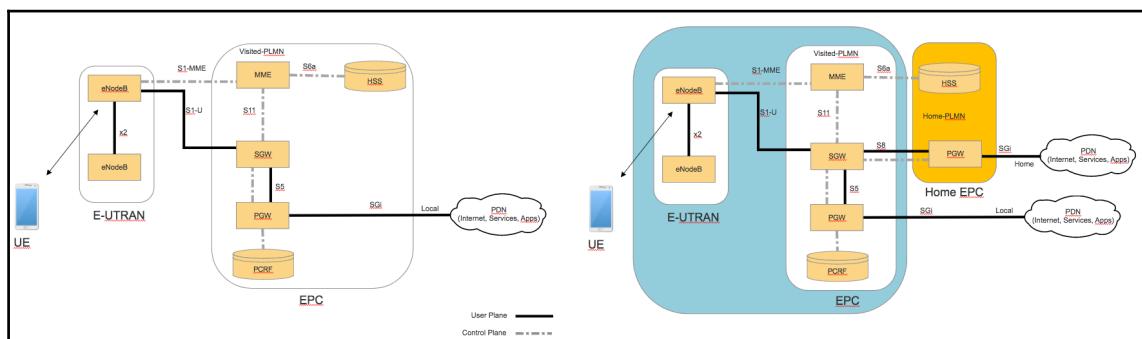
The typical 4G-LTE network has three components: a client, a radio network, and a core network. The client is simply the user's radio device. The radio network represents frontend communication between the client and the core network and includes radio equipment such as the tower. The core network represents the management and control interface of the carrier and may manage one or more radio networks.

The architecture can be decomposed as follows:

- **Evolved Universal Terrestrial Radio Access Network (E-UTRAN):** This is 4G-LTE air interface to LTE UE devices. E-UTRAN uses OFDMA for the downlink portion and SC-FDMA for the uplink. Subsequently, this makes it incompatible with the legacy 3G W-CDMA technology. The E-UTRAN consists of eNodeBs but can contain several that are interlinked by a channel called the **X2** interface.
- **eNodeB:** This is the core of the radio network. It handles communications between the UE and the core (EPC). Each eNodeB is a base station that controls eUEs in one or more cellular zones and allocates resources to a specific client in 1 ms chunks called **TTI**. It will allocate channel resources based on usage conditions to various UEs in its cell proximity. eNodeB systems also have the responsibility of triggering state transitions from IDLE to CONNECTED. It also handles the mobility of the UE such as the handover to other eNodeBs. It is also responsible for transmission and congestion control. The interface out of the eNodeB and into the EPC is the **S1** interface.
- **User Equipment (UE):** This is the client hardware and is composed of **Mobile Terminations (MT)**, which perform all the communication functions, **Terminal Equipment (TE)**, which manages terminating data streams, and the **Universal Integrated Circuit Card (UICC)**, which is the SIM card for identity management.
- **Evolved Packet Core (EPC):** In the design of LTE, the 3GPP decided to build a flat architecture, and separate user data (called the *user plane*) and control data (called the *control plane*). This is allowed for more efficient scaling. The EPC has five basic components listed here:
 - **Mobility Management Equipment (MME):** Responsible for control plane traffic, authentication and security, location and tracking, and mobility issue handlers. The MME also needs to recognize mobility in IDLE mode. This is managed using a **Tracking Area (TA)** code. The MME also governs the **Non-Access Stratum (NAS)** signaling and bearer control (described later).
 - **Home Subscriber Server (HSS):** This is a central database associated with the MME that contains information about the network operator subscribers. This may include keys, user data, maximum data rates on the plan, subscriptions, and so on. The HSS is a holdover from the 3G UMTS and GSM networks.
 - **Servicing Gateway (SGW):** Responsible for user plane and user data flow. Essentially, it acts as a router and forwards packets between the eNodeB and PGW directly. The interface out of the SGW is called the S5/S8 interface. S5 is used if two devices are on the same network and S8 used if they are on different networks.

- **Public Data Network Gateway (PGW):** Connects mobile networks to external sources including the internet or other PDN networks. It also allocates the IP address for the mobile devices connected. The PGW manages the **Quality of Service (QoS)** for various internet services such as video streaming and web browsing. It uses an interface called the *SGi* to reach into various external services.
- **Policy Control and Charging Rules Function (PCRF):** This is another database that stores policies and decision-making rules. It also controls the flow-based charging functions.
- **Public Data Network (PDN):** This is the external interface, and for the most part, the internet. It can include other services, data centers, private services, and so on.

In a 4G-LTE service, a customer will have a carrier or operator known as their **Public Land Mobile Network (PLMN)**. If the user is in that carrier's PLMN, he or she is said to be in *Home-PLMN*. If the user moves to a different PLMN outside their home network (for example, during international travel) then the new network is called the **visited-PLMN**. The user will connect their UE to a visited-PLMN which requires resources of the E-UTRAN, MME, SGW, and PGW on the new network. The PGW can grant access to a *local-breakout* (a gateway) to the internet. This, effectively, is where roaming charges start to affect the service plans. Roaming charges are applied by the visited-PLMN and accounted for on the client's bill. Below is a graphic illustrating this architecture. On the left side is a top-level view of the 3GPP System Architecture Evolution for 4G-LTE. In this case, it represents the client UE, the radio node E-UTRAN, and the core network EPC, all residing in a Home-PLMN. On the right is a model of the mobile client moving to a visited-PLMN and distributing the functionality between the visited-PLMN E-UTRAN and EPC, as well as crossing back to the home network. The S5 interconnect is used if the client and carrier reside on the same network and the S8 interface if the client is crossing across different networks.



Right: 3GPP System Architecture. Left: Top-Level View of 4G LTE Architecture.

Non-Access Stratum (NAS) signaling was mentioned in the MME. It is a mechanism for passing messages between UE and core nodes such as switching centers. Examples may include messages such as Authentication Messages, Updates, or Attach Messages. The NAS sits at the top of the SAE protocol stack.

The **GPRS Tunneling Protocol (GTP)** is an IP/UDP-based protocol used in LTE. GTP is used throughout the LTE communication infrastructure for control data, user data, and charging data. In the figure above, most of the S* channel's connecting components use GTP packets.

The LTE architecture and protocol stack use what is known as **bearers**. Bearers are a virtual concept used to provide a pipe to carry data from one node to another node. The pipe between the PGW and UE is called the **EPS Bearer**. As data enters the PGW from the internet, it will package the data in a GTP-U packet and send it to the SGW. The SGW will receive the packet, strip the GTP-U header, and repackage the user data in a new GTP-U packet on its way to the eNB. Again, the eNB repeats the process and will repack the user data after compressing, encrypting, and routing to logical channels. The message will then transmit to the UE through a radio bearer. One advantage bearers bring to LTE is QoS control. Using bearers, the infrastructure can guarantee certain bitrates dependent on customer, application, or usage.

When a UE attaches to a cellular network the first time, it is assigned a *default bearer*. Each default bearer has an IP address, and a UE may have several default bearers, each with a unique IP. This is a best-effort-service meaning it wouldn't be used for guaranteed QoS-like voices. A *dedicated bearer*, in this case, is used for QoS and good user experience. It will be initiated when the default bearer is incapable of fulfilling a service. The dedicated bearer always resides on top of the default bearer. A typical smartphone may have the following bearers running at any time:

- Default Bearer 1: Messaging and SIP signaling
- Dedicated Bearer: Voice data (VOIP) linked to default bearer 1
- Default Bearer 2: All smartphone data services such as email, browser

The switching aspect of LTE is also worth mentioning. We looked at the different generations of 3GPP evolutions from 1G to 5G. One goal of the 3GPP and carriers was to move to a standard and accepted **Voice over IP (VOIP)** solution. Not only would data be sent over standard IP interfaces but so would voice. After some contention between competing methods, the standards body settled on VoLTE as the architecture. VoLTE uses an extended variant of the **Session Initiation Protocol (SIP)** to handle voice and text messages. A codec known as the **Adaptive Multi-Rate (AMR)** codec provides wideband high-quality voice and video communication. Later, we will look at new 3GPP LTE categories that drop VoLTE to support IoT deployments.

It must be noted that LTE is one of two standards for mobile broadband.

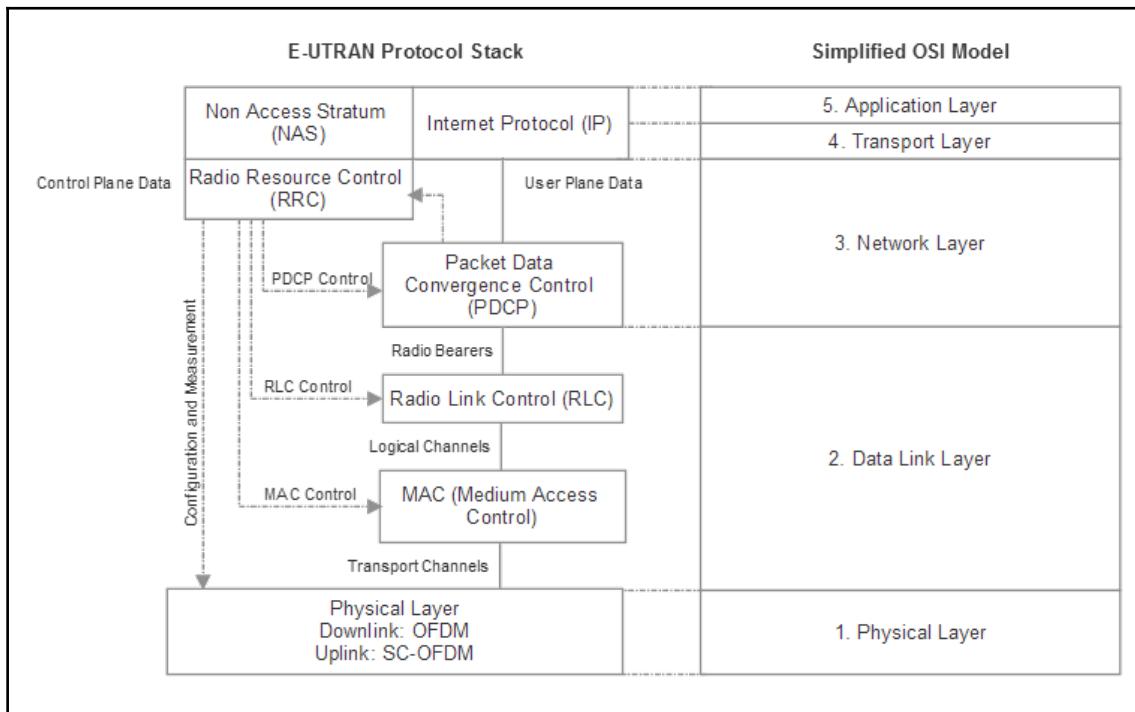
Wireless Mobility Internet Access (WiMAX) is the alternative. LTE WiMAX is a wideband, IP-based, OFDMA communication protocol. WiMAX is based on the IEEE802.16 standards and managed by the WiMAX Forum. WiMAX resides in the 2.3 and 3.5 GHz range of the spectrum but can reach into the 2.1 GHz and 2.5 GHz range, like LTE. Introduced commercially before LTE took off, WiMAX was the Sprint and Clearwire choice for high-speed data.



WiMAX, however, has only found niche uses. LTE is generally much more flexible and widely adopted. LTE also advanced slowly in an attempt to keep backward compatibility with older infrastructure and technologies in place, while WiMAX was meant for new deployments. WiMAX did have an advantage regarding ease of setup and installation over LTE, but LTE won the bandwidth race and in the end bandwidth defined the mobile revolution.

4G-LTE E-UTRAN protocol stack

The 4G-LTE stack has similar characteristics to other derivatives of the OSI model, however, there are other characteristics of the 4G control plane, as can be seen in the figure. One difference is that the **Radio Resource Control (RRC)** has extensive reach throughout the stack layers. This is called the **control plane**. This control plane has two states: *idle* and *connected*. When idle, the UE will wait in a cell after associating with it and may monitor a paging channel to detect if an incoming call or system information is targeted at it. In the connected mode, the UE will establish a downlink channel and information on the neighbor cell. The E-UTRAN will use the information to find the most suitable cell at the time:



E-UTRAN Protocol Stack for 4G-LTE, and comparison to the simplified OSI model.



The control plane and user plane also behave slightly differently and have independent latencies. The user plane typically has a 4.9 ms latency while the control plane will use a 50 ms latency.

The stack is composed of the following layers and features:

- **Physical Layer 1:** This layer is the radio interface also known as the *air interface*. Responsibilities include link adaption (AMC), power management, signal modulation (OFDM), digital signal processing, searching for cell signals, synchronization with cells, handover control, and cell measurements for the RRC layer.
- **Medium Access Control (MAC):** Similar to other OSI-derived stacks, this performs a mapping between logical channels and transport layers. The MAC layer multiplexes various packets onto **transport blocks (TB)** for the physical layer. Other responsibilities include scheduling reporting, error correction, channel prioritization, and managing multiple UEs.

- **Radio Link Control (RLC):** RLC transfers upper layer PDUs, performs error correction via ARQ, and handles the concatenation/segmentation of packets. Additionally, it provides a logical channel interface, detection of duplicate packets, and performs re-assembly of received data.
- **Packet Data Convergence Control Protocol (PDCP):** This layer is responsible for packet compression and decompression. The PDCP also manages the routing of user plane data as well as control plane data with feedback into the RRC. The management of duplicate SDUs (such as in a handover procedure) is handled in this layer. Other features include cipher and deciphering, integrity protection, timer-based discarding of data, and re-establishment of channels.
- **Radio Resource Control (RRC):** The RRC layer broadcasts system information to all the layers including **Non-Access Strata (NAS)** and **Access Strata (AS)**. It manages security keys, configurations, and the control of radio bearers.
- **Non-Access Stratum:** This represents the highest level of the control plane and is the main interface between a UE and the MME. The principal role is session management, therefore the mobility of the UE is established at this level.
- **Access Stratum:** This is a layer below the NAS whose purpose is to convey non-radio signals between the UE and radio network.

4G-LTE geographical areas, dataflow, and handover procedures

Before considering the process of cellular handover, we need to first define the vernacular surrounding areas and network identification. There are three types of geographical areas in an LTE network:

- **MME Pool Areas:** This is defined as a region where a UE can move without changing the serving MME.
- **SGW Service Areas:** This is defined as an area where one or more SGW will continue to provide service for a UE.
- **Tracking Areas (TA):** These define sub-areas composed of small MME and SGW areas that don't overlap. They are used to track the location of a UE that is in standby mode. This is essential for handover.

Each network in a 4G-LTE network must be uniquely identifiable for these services to work. To assist with identifying networks, the 3GPP uses a network ID consisting of:

- **Mobile Country Code (MCC):** Three digit identification of the country the network resides within (for example, Canada is 302)
- **Mobile Network Code (MNC):** A two or three digit value indicating the carrier (for example, Rogers Wireless is 720)

Each carrier also needs to uniquely identify each MME it uses and maintains. The MME is needed locally within the same network and globally when a device is moving and roaming to find the home network. Each MME has three identities:

- **MME Identifier:** This is the unique ID that will locate a particular MME within a network. It is composed of the following two fields:
 - **MME Code (MMEC):** This identifies all the MMEs that belong to the same pool area mentioned previously.
 - **MME Group Identity (MMEI):** This defines a group or cluster of MMEs.
- **Globally Unique MME Identifier (GUMMEI):** This is the combination of a PLMN-ID and the MMEI described previously. The combination forms an identifier that can be located anywhere globally from any network.

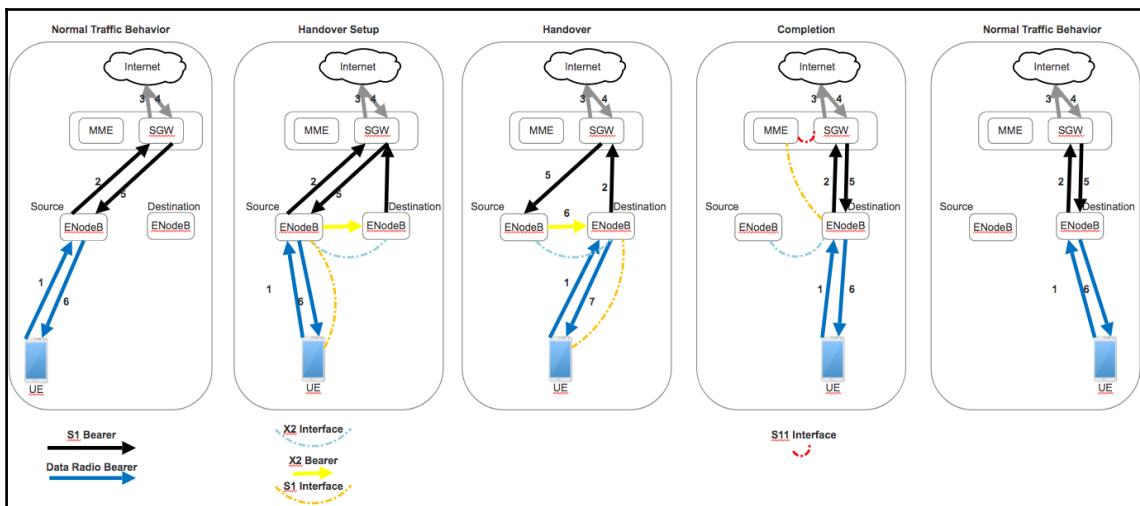
The **Tracking Area Identity (TAI)** allows a UE device to be tracked globally from any position. This is the combination of the PLMN-ID and the **Tracking Area Code (TAC)**. The TAC is a particular physical sub-region of a cell coverage area.

The cell ID is constructed from a combination of the **E-UTRAN Cell Identity (ECI)**, which identifies the cell in a network, the **E-UTRAN Cell Global Identifier (ECGI)**, which identifies the cell anywhere in the world, and a physical cell identity (integer value 0 to 503) used to distinguish it from another neighbor EU.

The handover process involves transferring a call or data session from one channel to another in a cell network. The handover most obviously occurs if the client is moving. A handover can also be instigated if the base station has reached its capacity and is forcing some devices to relocate to another base station in the same network. This is called **Intra-LTE Handover**. Handover can also occur between carriers while roaming, called **Inter-LTE Handover**. It is also possible to hand over to other networks (Inter-RAT Handover), such as moving between a cellular signal and a Wi-Fi signal.

If the handover exists in the same network (Intra-LTE Handover) then two eNodeBs will communicate through the X2 interface and the core network EPC is not involved in the process. If the X2 isn't available, the handover needs to be managed by the EPC through the S1 interface. In each case, the source eNodeB initiates the transaction request. If the client is performing an Inter-LTE, handover is more complicated as two MMEs are involved: a source (S-MME) and a target (T-MME).

The process allows for seamless handover, as shown in the series of steps in the figure here. First, the source eNodeB will decide to instantiate a handover request based on capacity or client movement. The eNodeB does this by broadcasting a MEASUREMENT CONTROL REQ message to the UE. These are network measurement reports sent out when certain thresholds are hit. A **Direct Tunnel Setup (DTS)** is created where the X2 transport bearer communicates between the source eNodeB and the destination eNodeB. If the eNodeB determines it is appropriate to launch a handover, it finds the destination eNodeB and sends a RESOURCE STATUS REQUEST over the X2 interface to determine if the target can accept a handover. The handover is launched with the HANDOVER REQUEST message. The destination eNodeB prepares resources for the new connection. The source eNodeB will detach the client UE. Direct packet forwarding from the source eNodeB to the destination eNodeB ensures no packets in flight are lost. Next, the handover completes by using a PATH SWITCH REQUEST message between the destination eNodeB and the MME to inform it that the UE has changed cells. The S1 bearer will be released, as will the X2 transport bearer, because no residue packets should be flowing to the EU client at this time:



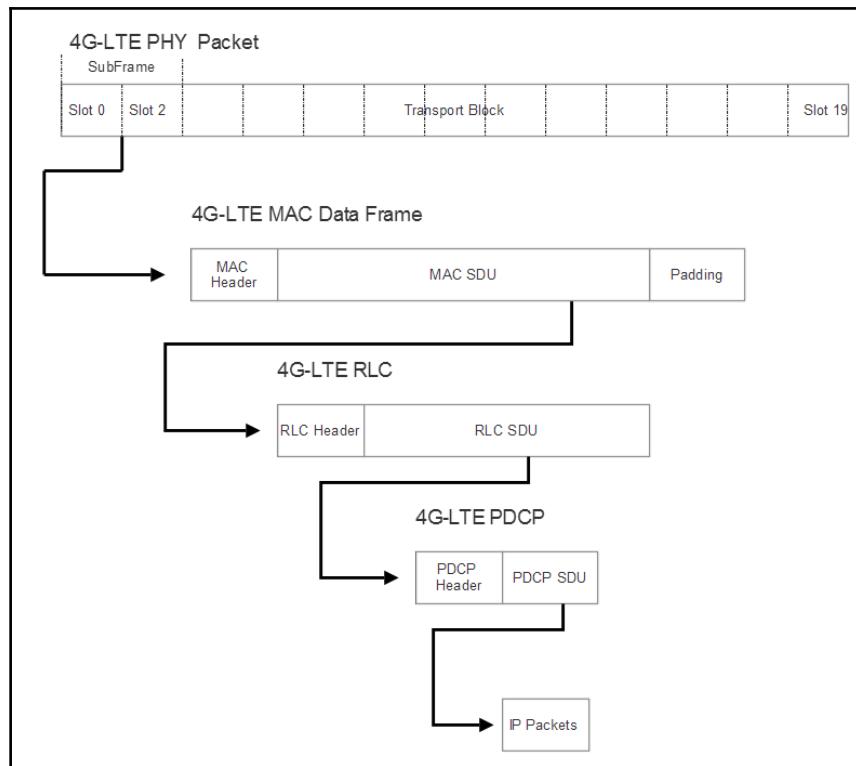
An example of Intra-LTE handover between two eNodeBs.



A number of IoT gateway devices using 4G-LTE allow for multiple carriers (such as Verizon and ATT) to be available on a single device gateway or router. Such gateways can switch and handover between carriers seamlessly without data loss. This is an important characteristic for mobile and transportation based IoT systems such as logistics, emergency vehicles, and asset tracking. Handover allows such moving systems to migrate between carriers for better coverage and efficient rates.

4G-LTE packet structure

The LTE packet frame structure is similar to other OSI models. The figure here illustrates the decomposition of the packet from **PHY** up to the **IP** layer. The IP packet is enveloped in the 4G-LTE layers:



4G-LTE Packet Structure. IP packets are reformed in the PDCP SDUs and flow through the RLC, MAC, and PHY layers. The PHY creates slots and sub-frames of transport blocks from the MAC layer.

Cat 0, Cat 1, Cat M1, and NB-IoT

The connectivity of an IoT device to the internet is different from typical consumer-based cellular devices like a smartphone. A smartphone mainly pulls information off the internet in a downlink. Often that data is large and streaming in real time, such as video data and music data. In an IoT deployment, the data can be very sparse and arrive in short bursts. More often than not, the majority of data will be generated by the device and travel over the uplink. The LTE evolution has progressed in building a cellular infrastructure and a business model focused and optimized for mobile consumers. The new shift is to satisfy the IoT producers of data at the edge as that will dwarf the number of consumers. The following sections cover **Low Power Wide Area Networks (LPWAN)** and specifically the LTE variety of LPWAN. These are suitable for IoT deployments and the features vary.

Up until release 13, the lowest data rate suitable for typical IoT devices was Cat-1. As the mobile revolution demanded higher speeds and services, Cat-1 was overlooked in the 3G and 4G timeline. Release 12 and 13 addressed the IoT device requirements for low cost, low power, sparse transmission, and range extensions.



One thing all these protocols share by design is that they are all compatible with the existing cellular hardware infrastructure. However, to enable new features, software changes to the protocol stack are necessary for the infrastructure. Without such changes, Cat-0, Cat-M1, and Cat-NB UEs will not even see the network. The IoT architect needs to ensure that the cellular infrastructure they intend to deploy into has been updated to support these standards.

Cat-1 hasn't gained significant traction in the market and we won't go into the specification here as it is similar to the 4G-LTE material discussed previously.

LTE Cat-0

Cat-0 was introduced in release 12 and was the first architecture outside of Cat-1 targeted to IoT needs. The design, like many other LTE specifications, is IP-based and runs in the licensed spectrum. A significant difference is in the uplink and downlink peak data rates (1 Mbps each), as opposed to Cat-1 at 10 Mbps for the downlink and 5 Mbps for the uplink. While the channel bandwidth stays at 20 MHz, the reduction in data rates greatly simplifies the design and reduces cost. Additionally, moving from a full duplex to a half-duplex architecture further improves cost and power.

Usually, the NAS layer of the LTE stack doesn't have much of a role in servicing UEs. In Cat-0, the 3GPP architects changed the NAS abilities to assist in power saving at the UE level. Cat-0 introduces **Power Save Mode (PSM)** to the LTE specification to address hard power constraints. In a traditional LTE device, the modem remains connected to a cellular network, consuming power even if the device is active, idle, or sleeping. To prevent connection power overhead a device can disconnect and disassociate from a network, but it would incur a 15 to 30 second reconnect and search phase. PSM allows the modem to enter a very deep sleep state—at any time it is not actively communicating—yet wake quickly. It does this by performing a **Tracking Area Update (TAU)** periodically and remaining reachable via paging for a programmable period of time. Essentially, an IoT device could enter an idle period of 24 hours and wake once a day to broadcast sensor data all while staying connected. All the negotiations for setting these individual timers are managed via the changes at the NAS level and are relatively simple to enable by setting two timers:

- **T3324:** Time the UE stays in idle mode. If the IoT device application is certain no messages are pending, it can reduce the value of the timer.
- **T3412:** Once the T3324 timer expires, the device will enter PSM for a period of T3412. The device will be at the lowest possible energy state. It cannot participate in paging or network signaling. The device, however, maintains all UE state (bearers, identities). The maximum time allowed is 12.1 days.



When using PSM or other advanced power management modes, it is wise to test compliance with carrier infrastructure. Some cell systems require a ping every 2 to 4 hours to the UE. If the carrier loses connectivity for periods greater than 2 to 4 hours, they may deem the UE unreachable and detach.

There is little coverage and adoption of Cat-0, and it has seen limited growth. Most of the new features of Cat-0 have been included in Cat-1 and other protocols.

LTE Cat-1

LTE Cat-1 reuses the same chipsets and carrier infrastructure as Cat-4, so it can be used throughout the USA on Verizon and AT&T infrastructure. Cat-1 has significant market traction in the M2M industry. The specification was part of release eight and was later updated to support power saving mode and the single LTE antenna of Cat-0.

Cat-1 is considered a medium-speed LTE standard, meaning the downlink is 10 Mbps and the uplink is 5 Mbps. At these rates, it is still capable of transporting voice and video streams as well as M2M and IoT data payloads. By adopting the Cat-0 PSM and antenna design, Cat-1 operates at a with less power than traditional 4G-LTE. It is also significantly less costly to design radios and electronics.

Cat-1 is currently the best choice in cellular connectivity for IoT and M2M devices with the widest coverage and lowest power. While significantly slower than regular 4G-LTE (10 Mbps versus 300 Mbps downlink), the radio can be designed to fall back to 2G and 3G networks as needed. Cat-1 reduces design complexity by incorporating time-slicing, which also reduces the speed considerably.

Cat-1 can be thought of as a complement to newer narrowband protocols, covered next.

LTE Cat-M1 (eMTC)

Cat-M1, also known as enhanced Machine-Type Communication (and sometimes just called Cat-M), was designed to target IoT and M2M use cases with low cost, low power, and range enhancements. Cat-M1 was released in the 3GPP release 13 schedule. The design is an optimized version of the Cat-0 architecture. The single largest difference is that the channel bandwidth is reduced from 20 MHz to 1.4 MHz. Reducing the channel bandwidth from a hardware point of view relaxes timing constraints, power, and circuitry. Costs are also reduced by up to 33% compared to Cat-0 since the circuitry doesn't need to manage a wide 20 MHz spectrum. The other significant change is in the transmit power, which is reduced from 23 dB to 20 dB. Reducing the transmit power by 50% reduces cost further by eliminating an external power amplifier and enables a single-chip design. Even with the reduction in transmit power, the coverage improves by +20 dB.

Cat-M1 follows other late 3GPP protocols that are IP-based. While not a MIMO architecture, throughput is capable of 375 Kbps or 1 Mbps on the uplink as well as downlink. The architecture allows for mobility and justifiably for in-vehicle or V2V communication. The bandwidth is wide enough to allow for voice communication as well, using VoLTE. Multiple devices are allowed on a Cat-M1 network using the traditional SC-FDMA algorithm. Cat-M1 also makes use of more complex features such as frequency hopping and turbo-coding.

Power is critical in IoT edge devices. The most significant power reduction feature of Cat-M1 is the transmission power change. As mentioned, the 3GPP organization reduced the transmission power from 23 dB to 20 dB. This reduction in power does not necessarily mean a reduction in range. The cell towers will rebroadcast packets six to eight times. This is to ensure reception in particularly problematic areas. Cat M1 radios can turn off reception as soon as they receive an error-free packet.

Another power saving feature is the **Extended Discontinuous Receive (eDRX)** mode which allows for a sleep period of 10.24 seconds between paging cycles. This reduces power considerably and enables the UE to sleep for a programmable number of **hyper-frames (HF)** of 10.24 seconds each. The device can enter this extended sleep mode for up to 40 minutes. This allows the radio to have an idle current as low as 15 uA.

Additional power mitigation abilities and features include:

- PSM, as introduced in Cat-0 and release 13.
- Relaxing neighbor cell measurements and reporting periods. If the IoT device is stationary or slow moving (sensor on a building, cattle in a feedlot) then the call infrastructure can be tuned to limit the control messages.
- User and control plane CIoT EPS optimization is a feature that is part of the RRC in the E-UTRAN stack. In normal LTE systems, a new RRC context must be created every time the UE wakes from IDLE mode. This consumes the majority of power when the device simply needs to send a limited amount of data. Using the EPS optimization, the context of the RRC is preserved.
- Header compression of TCP or UDP packets.
- Reduction of sync time after long sleep periods.

The next section covers Cat-NB. The market has developed a perception that Cat-NB is substantially lower in power and cost over all other protocols such as Cat-M1. That is partially true, and an IoT architect must understand the use cases and carefully choose which protocol is correct. For example, if we hold the transmit power constant between Cat-NB and Cat-M1, we see Cat-M1 has an 8 dB gain in coverage. Another example is power; while Cat-M1 and Cat-NB have similar and aggressive power management features, Cat-M1 will use less power for larger data sizes. Cat-M1 can transmit the data faster than Cat-NB and enter a deep sleep state that much sooner. This is the same concept Bluetooth 5 used to reduce power, simply by sending the data faster and then entering a sleep state. Also, as of the time of writing, Cat-M1 is available today while Cat-NB has not hit USA markets.



LTE Cat-NB

Cat-NB, also known as NB-IoT, NB1, or Narrow Band IoT, is another LPWAN protocol governed by the 3GPP in release 13. Like Cat-M1, Cat-NB operates in the licensed spectrum. Like Cat-M1, the goal is to reduce power (10-year battery life), extend coverage (+20 dB), and reduce cost (\$5 per module). Cat-NB is based on the **Evolved Packet System (EPS)** and optimizations to the **Cellular Internet of Things (CIoT)**. Since the channels are much narrower than even the 1.4 MHz Cat-M1, cost and power can be reduced even further with the simpler designs of the analog-to-digital and digital-to-analog converters.

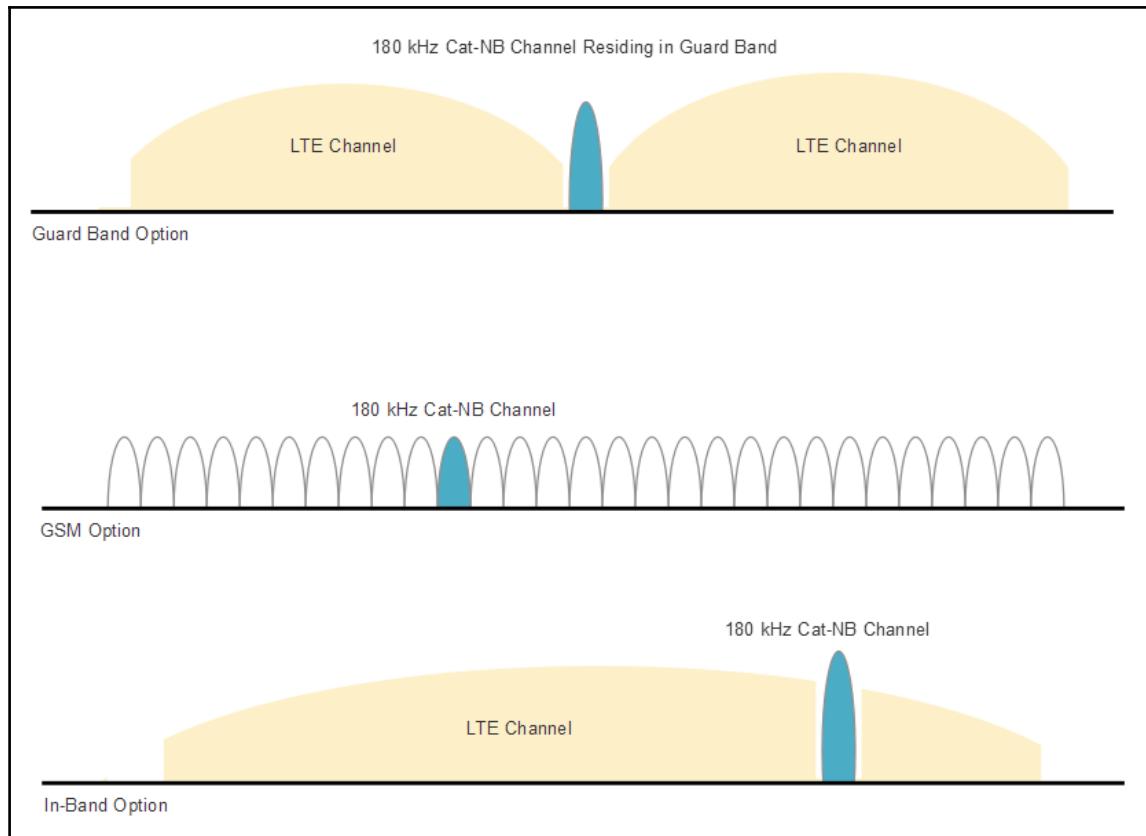
Significant differences between Cat-NB and Cat-M1 include:

- **Very narrow channel bandwidth:** As with Cat-M1, which reduced the channel width to 1.4 MHz, Cat-NB reduces it further to 180 kHz for the same reasons (reducing cost and power).
- **No VoLTE:** As the channel width is so low, it does not have the capacity for voice or video traffic.
- **No mobility:** Cat-NB will not support handover and must remain associated with a single cell or remain stationary. This is fine for the majority of secured and fastened IoT sensor instruments. This includes handover to other cells and other networks.

Regardless of these significant differences, Cat-NB is based on OFDMA (downlink) and SC-FDMA (uplink) multiplexing and uses the same subcarrier spacing and symbol duration. The E-UTRAN protocol stack is also the same with the typical RLC, RRC, and MAC layers, and remains IP-based but is considered a new air interface for LTE.

Since the channel width is so small (180 kHz), it allows for the opportunity to *bury* the Cat-NB signal inside a larger LTE channel, replace a GSM channel, or even exist in the guard channel of regular LTE signals. This allows for flexibility in LTE, WCDMA, and GSM deployments. The GSM option is simplest and fastest to market. Some portions of the existing GSM traffic can be placed on the WCDMA or LTE network. That frees up a GSM carrier for IoT traffic. In-band provides a massive amount of spectrum to use as the LTE bands are much larger than the 180 kHz band. This allows for deployments of up to 200,000 devices in theory per cell. In this configuration, the base cell station will multiplex LTE data with Cat-NB traffic. This is entirely possible since the Cat-NB architecture is a self-contained network and interfaces cleanly with existing LTE infrastructure. Finally, using Cat-NB as the LTE guard band is a unique and novel concept. Since the architecture re-uses the same 15 kHz subcarriers and design, it can be accomplished with existing infrastructure.

The following figure illustrates where the signal is allowed to reside:



Cat-NB deployment options as a guard band, within a GSM signal, or in-band with LTE.

Since the **Maximum Coupling Loss (MCL)** is 164 dB, it allows for deep coverage in basements, tunnels, rural areas, and open environments. The 20 dB improvement over standard LTE is a 7x increase in the coverage area. The data rate obtainable is a function of the SNR and the channel bandwidth, as we have seen in Shannon-Hartley's theorem. For the uplink communication, Cat-NB will assign each UE one or more 15 kHz subcarriers in a 180 kHz resource block. Cat-NB has the option to reduce the subcarrier width to as low as 3.75 kHz, which allows more devices to share the space. However, one must carefully inspect the interference potential between edge 3.75 kHz subcarriers and the next 15 kHz subcarrier.

The data rate, as we have learned, is a function of coverage. Ericsson has conducted tests illustrating the effect of varying coverage and signal strength. The data is revealing and explains why latency may become a significant issue for Cat-NB.



At the cell border: Coverage = +0dB, Uplink Time = 39ms, Total Transmit Time = 1,604 ms.

At medium coverage: Coverage = +10dB, Uplink Time = 553 ms, Total Transmit Time = 3,085 ms.

Worst case: Coverage: +20dB, Uplink Time = 1,923 ms, Total Transmit Time = 7,623 ms.

From: "NB-IOT: a sustainable technology for connecting billions of devices", Ericsson Technology Review Volume 93, Stockholm, Sweden, #3 2016

Power management is very similar to Cat-M1. All the power management techniques in release 12 and 13 apply to Cat-NB as well (PSM, eDRX, and all the other features are included).

5G

5G (or 5G-NR for new radio) is the next generation IP-based communication standard being drafted and designed to replace 4G-LTE. That said, it draws on some technologies of 4G-LTE but comes with substantial differences and new capabilities. So much material has been written on 5G that it dwarfs even current Cat-1 and Cat-M1 material. In part, 5G promises to deliver substantial abilities for IoT, commercial, mobile, and vehicular use cases. 5G also improves bandwidth, latency, density, and user cost. Rather than build different cellular services and categories for each use case, 5G attempts to be a single umbrella standard to serve them all. Meanwhile, 4G-LTE will continue to be the predominant technology for cellular coverage and will continue to evolve. 5G is not a continuing evolution of 4G; it derives from 4G but is a new set of technologies. In this section, we will only discuss elements that pertain to IoT use cases or are of merit and have the potential to become part of the 5G specification.

5G is aiming for initial customer launch in 2020, however, mass deployment and adoption may follow years later in the mid-2020s. The goals and architecture of 5G are still evolving, and have been since 2012. There have been three distinct and different goals for 5G: <http://www.gsmhistory.com/5g/>. Goals range from converged fiber and cellular infrastructure, ultra-fast mobiles using small cells, and lowered cost barriers of mobile, all part of the 5G story. Again, the ITU-R has approved the international specifications and standards, while the 3GPP is following with a set of standards to match the ITU-R timeline. The 3GPP RAN has already begun analyzing study items as of release 14. The intent is to produce a two-phase release of 5G technologies. Phase one will complete in 2018 with release 15 and phase two is targeted for release 19 in 2019. Both align with the 2020 goal of commercial launch.



Verizon is planning a commercial deployment of 5G by mid-2018 (pre-3GPP specification). 3GPP has accelerated 5G as **non-standalone (NSA)** and **standalone (SA)**. NSA will use the LTE core while SA will use the 5G Next Generation Core. 5G for IoT-specific use cases is several years out since the initial market focus is mobile and smartphone users.

The general consensus of features in 5G that should be understood and have a good prospect of being in the final release are:

1. Enhanced Mobile Broadband (eMBB):

- 1 to 10 GBps connections to UEs/endpoints in the field (not theoretical)
- 100% coverage across the globe (or perception of)
- 10 to 100x the number of connected devices over 4G-LTE
- Connectivity at a speed of 500 km/h

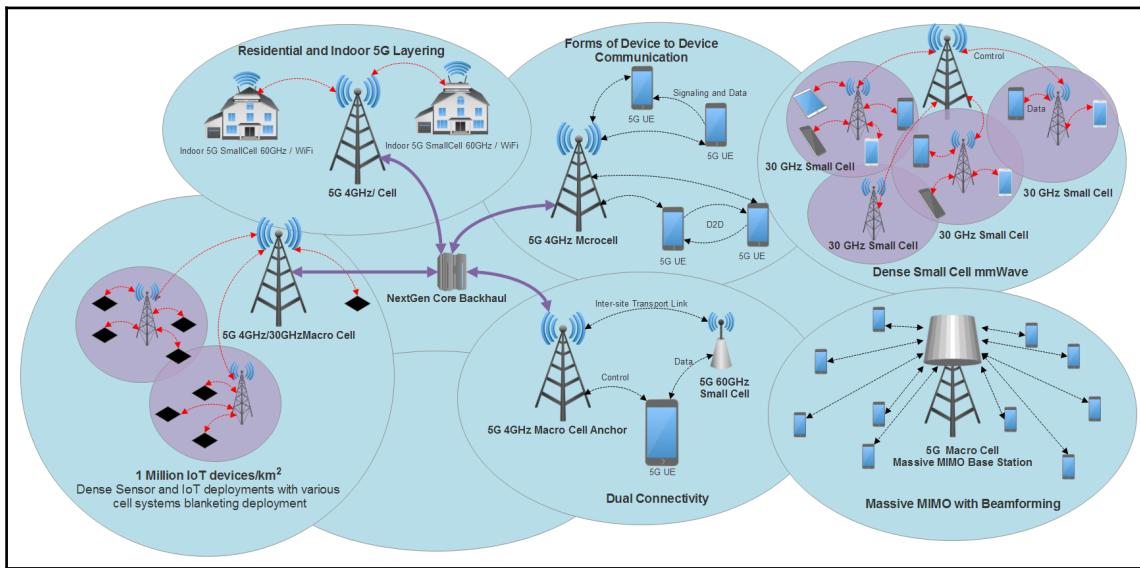
2. Ultra-Reliable and Low-Latency Communications (URLLC):

- Sub < 1 ms end-to-end round-trip latency
- 99.999% availability (or perception of)

3. Massive Machine Type Communications (mMTC):

- 1000x bandwidth per unit area; this implies roughly 1 million nodes in 1 km²
- Up to a 10 year battery life on endpoint IoT nodes

- 90% reduction in network energy usage:



5G Topologies. From left to right: 1 million node density through small cells and macrocell deployment. Indoor and home use of 60 Ghz with macrocell at 4 Ghz backhaul. Dual connectivity example with split control and data planes using two radios for user data and 4 GHz macrocells for the control plane. Device to device connectivity. Massive MIMO with beamforming from a single mmWave antenna. Density increase with a mix of small cells in mmWave for blanket coverage of user data.

Current 4G-LTE systems mainly use frequencies below the 3 GHz range. 5G will radically change the spectrum usage. While the space below 3 GHz is significantly congested and sliced into slivers of bandwidth, 5G may use a multitude of frequencies. Under strong consideration is the use of **millimeter waves (mmWave)** in the unlicensed 24 to 100 GHz range. These frequencies directly address Shannon's Law by increasing the bandwidth B of the law with extremely wide channels. Since the mmWave space is not saturated or sliced up by various regulatory bodies, channels as wide as 100 MHz in the 30 GHz to 60 GHz frequencies are possible. This will provide the technology to support multi-gigabit per second speeds.

The principal issues with mmWave technology are free space path loss, attenuation, and penetration. If we remember that free space path loss can be calculated as $L_{fs} = 32.4 + 20\log_{10}f + 20\log_{10}R$ (where f is the frequency and R is the range), then we can see how the loss is affected by a 2.4, 30, and 60 GHz signal as:

- 2.4 GHz, 100 m range: 80.1 dB
- 30 GHz, 100 m range: 102.0 dB
- 60 GHz, 100 m range: 108.0 dB
- 2.4 GHz, 1 km range: 100.1 dB
- 30 GHz, 1 km range: 122.0 dB
- 60 GHz, 1 km range: 128.0 dB

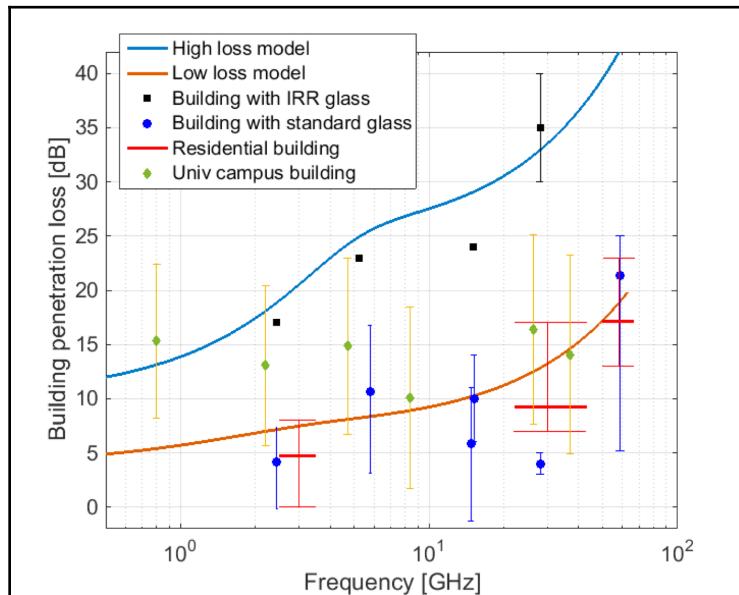
20 dB is significant, but with mmWave, antennas can accommodate significantly more antenna elements than a 2.4 GHz antenna. Free path loss is significant only if the antenna gain is independent of frequency. If we keep the antenna area constant, it is possible to mitigate the effects of path loss. This requires **Massive-MIMO (M-MIMO)** technology. M-MIMO will incorporate macrocell towers with 256 to 1024 antennas. Beamforming at the macrocell will be used as well. When combined with mm-Waves, M-MIMO has challenges in terms of contamination from nearby towers, and multiplexing protocols like TDD will need to be re-architected.



Another challenge with 5G is the need for very large antenna arrays to support M-MIMO with hundreds of antennas in a dense tower configuration. Under consideration are tightly packed 3D structures of antennas to support beamforming at the tower. Factors such as wind and storm effects on these towers still need to be resolved.

Attenuation is a very significant issue. At 60 GHz, a signal will be absorbed by oxygen in the atmosphere. Even vegetation and the human body itself will have a serious effect on signals. A human body will absorb so much RF energy at 60 GHz that it will form a shadow. A signal traveling at 60 GHz will have a loss of 15 dB/km. Thus, long-range communication will be subpar using 5G at 60 GHz and will require either a blanket of small cells or a drop to a slower frequency space. This is one of the reasons the 5G architecture will require multiple bands, small cells, macrocells, and a heterogeneous network.

Finally, material penetration in the mmWave spectrum is challenging. mmWave signals attenuate through drywall at 15 dB and glass windows on buildings contribute to a 40 dB loss. Therefore, indoor coverage with a macrocell is close to impossible. This and other types of signaling issues will be mitigated through the widespread use of indoor small cells:



Various Frequencies Versus Penetration Loss (dB). Typical building composite materials were tested (glass, brick, wood) from external to internal areas. The loss of infrared reduction glass is particularly difficult for mm-Wave frequencies. Source: Aalto University et al., “5G channel model for bands up to 100 GHz,” 3rd Workshop on Mobile Communications in Higher Frequency Bands (MCHFB), White Paper, Dec. 2016.

UEs may use multiple bands simultaneously. For example, an endpoint device may use lower frequencies for long-range communication and switch to mmWave for indoor and personal communication. Another scheme being considered is *Dual Connectivity*. Dual Connectivity steers data traffic across multiple bands dependent on data type. For example, the control plane and user plane of the stack are already divided. User plane data may steer to a nearby small cell using a 30 GHz frequency while control data may be steered to a long-range eNodeB macrocell tower at the typical 4 Ghz rate.

Another improvement for increased speed is spectral efficiency. The 3GPP is focusing on certain design rules:

- 15 kHz subcarrier spacing to improve multiplexing efficiency.
- Flexible and scalable numerology of symbols from 2^M symbols to 1 symbol to reduce latency.

As previously discussed, spectral efficiency is given as bps/Hz, using D2D and M-MIMO to improve spectral efficiency along with changes to the air interface and new radio. 4G-LTE uses OFDM, which works well for large data transmissions. However, for IoT and mMTC, the packets are much smaller. The overhead for OFDM also impacts latency in very dense IoT deployments. Hence, new waveforms are being architected for consideration:

- **Non-orthogonal Multiple Access (NOMA):** Allows multiple users to share a wireless medium.
- **Filter Bank Multi-Carrier (FBMC):** Controls the shape of subcarrier signals to remove side-lobes through DSPs.
- **Sparse Coded Multiple Access (SCMA):** Allows data to be mapped to different code from different codebooks.

Latency reduction is also a goal of the ITU and 3GPP. Latency reduction is crucial for 5G use cases such as interactive entertainment and virtual reality headsets, but also for industrial automation. However, it plays a large role in power reduction (another ITU goal). 4G-LTE can have a latency of up to 15 ms on 1 ms subframes. 5G is preparing for sub-1 ms latency. This too will be accomplished using small cells to route rather than congested macrocells. The architecture also plans for **Device to Device (D2D)** communication, essentially taking the cell infrastructure out of the data path for communication between UEs.

4G systems will continue to exist, as the rollout of 5G will take several years. There will be a period of coexistence that needs to be established. Release 15 will add further definitions to the overall architecture such as channel and frequency choices. From an IoT architect perspective, 5G is a technology to watch and plan for. IoT devices may predicated a WAN that will need to operate for a dozen years or more in the field. For a good perspective on the key points, constraints, and the detailed design of 5G, readers should refer to *5G: A Tutorial Overview of Standards, Trials, Challenges, Deployment, and Practice*, by M. Shafi et al., in *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 6, pp. 1201-1221, June 2017.

LoRa and LoRaWAN

LPWAN also include technologies that are proprietary and not sponsored by 3GPP. It can be argued that some IEEE802.11 protocols should also be classified in LPWAN, but the next two sections will highlight LoRa and Sigfox. LoRa is a physical layer for a long-range and low-power IoT protocol while LoRaWAN represents the MAC layer.



These proprietary LPWAN technologies and carriers have the advantage of using the unlicensed spectrum and that, simply, is data plan cost.

Typically, technologies like Sigfox and LoRaWAN will be 5x to 10x lower in their data rates compared to traditional 3G or LTE connections for large volume deployments (>100,000 units). That may change with more competition from Cat-M1, Cat-NB, and Cat-5 but it is too early to tell.

The architecture was originally developed by Cycleo in France but then acquired by the Semtech Corporation (a French mixed-signal electronics manufacturer) in 2012 for \$5 million in cash. The LoRa Alliance was formed in March of 2015. The alliance is the standard body for the LoRaWAN specification and technology. They also have a compliance and certification process to ensure interoperability and conformity to the standard. The alliance is supported by IBM, Cisco, and over 160 other members.

LoRaWAN has gained traction in Europe with network deployments by KPN, Proximus, Orange, Bouygues, Senet, Tata, and Swisscom. Other areas have sparse coverage to date.



One reason for the price difference, besides being in the unlicensed spectrum, is that a single LoRaWAN gateway has the potential to cover a significant amount of area. Belgium, with a land area of 30,500 km², is completely covered by seven LoRaWAN gateways. Typical range is 2 to 5 km in urban areas and 15 km in suburban areas. This reduction in infrastructure cost is very different than 4G-LTE with much smaller cells.

Since LoRa is the bottom of the stack, it has been adopted in competing architectures to LoRaWAN. Symphony Link, for example, is Link Labs's LPWAN solution based on the LoRa PHY, using an eight-channel, sub-GHz base station for industrial and municipal IoT deployments. Another competitor using LoRa is Haystack, who produces the DASH7 system. DASH7 is a full network stack on the LoRa PHY (not just the MAC layer).

The next section will focus exclusively on LoRaWAN.

LoRa physical layer

LoRa represents the physical layer of a LoRaWAN network. It manages the modulation, power, receiver and transmission radios, and signal conditioning.

The architecture is based on the following bands in ISM license-free space:

- **915 MHz:** In the USA with power limits but no duty cycle limit.
- **868 MHz:** In Europe with a 1% and 10% duty cycle.
- **433 MHz:** In Asia.

A derivative of **Chirp Spread Spectrum (CSS)** is the modulation technique used in LoRa. CSS balances data rate with sensitivity in a fixed channel bandwidth. CSS was first used in the 1940s for military long-range communication by using modulated *chirp* pulses to encode data and was found to be especially resilient against interference, Doppler effects, and multipath. Chirps are sinusoidal waves that increase or decrease over time. Since they use the entire channel for communication they are relatively robust in terms of interference. We can think of chirps signals with increasing or decreasing frequencies (sounding like a whale call). The bitrate which LoRa is a function of the chirp rate and the symbol rate. The bitrate is represented by R_b , the spreading factor by S , the bandwidth by B . Therefore the bitrate (bps) can range from 0.3 kbps to 5 kbps and is derived as:

$$R_b = S \times \frac{1}{\left[\frac{2^S}{B} \right]}$$

This form of modulation allows for low power over long distances, as the military found. Data is encoded using the increasing or decreasing frequency rate and multiple transmissions can be sent with different data rates on the same frequency. CSS allows signals to be received at 19.4 dB below the noise floor using FEC. The band is also subdivided into multiple sub-bands. LoRa uses 125 kHz channels and dedicates six 125 kHz channels and pseudo-random channel hopping. A frame will be transmitted with a specific spreading factor. The higher the spreading factor, the slower the transmission but the longer the transmission range. The frames in LoRa are orthogonal, meaning multiple frames can be sent simultaneously as long as each is sent with a different spreading factor. In total there are six different spreading factors (SF = 7 to SF = 12).

The typical LoRa packet contains a preamble, a header, and a 51 to 222-byte payload.

LoRa networks have a powerful feature called **Adaptive Data Rate (ADR)**. Essentially this enables dynamically scalable capacity based on the density of nodes and infrastructure. ADR is controlled by the network management in the cloud. Nodes that are close to a base station can be set to a higher data rate because of signal confidence. Those nodes in close proximity can transmit data, release their bandwidth, and enter a sleep state quickly versus distant nodes which transmit at slower rates.

The table below describes the uplink and downlink features:

Feature	Uplink	Downlink
Modulation	CSS	CSS
Link Budget	156 dB	164 dB
Bit Rate (Adaptive)	0.3 to 5 kbps	0.3 to 5 kbps
Message Size Per Payload	0 to 250 bytes	0 to 250 bytes
Message Duration	40 ms to 1.2 s	20 to 160 ms
Energy Spent Per Message	$E_{tx} = 1.2s * 32 \text{ mA} = 11 \text{ uAh}$ at full sensitivity $E_{tx} = 40 \text{ ms} * 32 \text{ mA} = 0.36 \text{ uAh}$ at min sensitivity	$E_{tx} = 160 \text{ ms} * 11 \text{ mA} = 0.5 \text{ uAh}$

LoRaWAN MAC layer

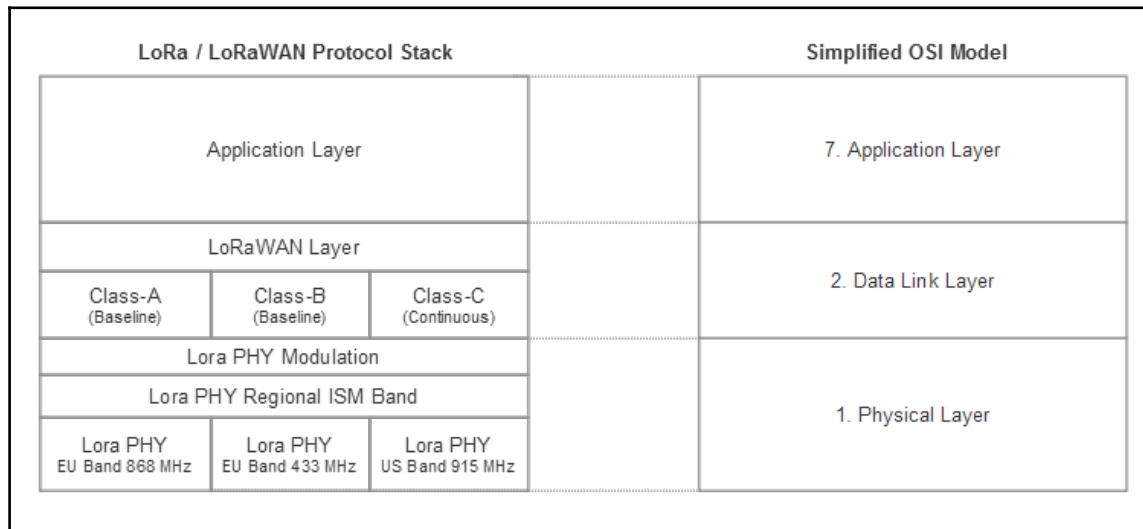
LoRaWAN represents the MAC that resides on top of a LoRa PHY. The LoRaWAN MAC is an open protocol while the PHY is closed. There are three MAC protocols that are part of the data link layer. These three balance latency with energy usage. Class-A is the best for energy mitigation while having the highest latency. Class-B is between Class-A and Class-C. Class-C has minimum latency but the highest energy usage.

Class-A devices are battery based sensors and endpoints. All endpoints that join a LoRaWAN network are first associated as Class-A with the option of changing class during operation. Class-A optimizes power by setting various *Receive Delays* during transmission. An endpoint starts by sending a packet of data to the gateway. After the transmission, the device will enter a sleep state until the *Receive Delay* timer expires. When the timer expires, the endpoint wakes and opens a receive slot and awaits a transmission for a period of time and then re-enters the sleep state. The device will wake again as another timer expires. This implies that all downlink communication occurs during a short period of time after a device sends a packet uplink. That period can be an extremely long time.

Class-B devices balance power and latency. This type of device relies on a beacon being sent by the gateway at regular intervals. The beacon synchronizes all endpoints in the network and is broadcast to the network. When a device receives a beacon, it creates a ping slot which is a short reception window. During these ping slot periods, messages can be sent and received. At all other times, the device is in a sleep state. Essentially this is a gateway-initiated session and based on a slotted communication method.

Class-C endpoints use the most power but have the shortest latency. These devices open two Class-A receive windows as well as a continuously powered receive window. Class-C devices are usually powered on and may be actuators or plug-in devices. There is no latency for the downlink transmission. Class-C devices cannot implement Class-B.

The LoRa/LoRaWAN protocol stack can be visualized as follows:



LoRa and LoRaWAN Protocol Stack. Comparison to standard OSI model. Note: LoRa/LoRaWAN represents only layer 1 and layer 2 of the stack model.

LoRaWAN security encrypts data using the AES128 model. One difference in its security from other networks is LoRaWAN separates authentication and encryption. Authentication uses one key (NwkSKey), and user data a separate key (AppSKey). To join a LoRa network, devices will send a JOIN request. A gateway will respond with a device address and authentication token. The application and network session keys will be derived during the JOIN procedure. This process is called the **Over-the-Air-Activation (OTAA)**. Alternatively, a LoRa-based device can use *Activation by Personalization*. In this case, a LoRaWAN carrier/operator preallocates 32-bit network and session keys and a client will purchase a connectivity plan and appropriate set of keys. The keys will be ordered from an endpoint manufacturer with the keys burnt into the device.

LoRaWAN is an asynchronous, ALOHA-based protocol. The pure ALOHA protocol was initially devised at the University of Hawaii in 1968 as a form of multiple access communication before technologies such as CSMA existed. In ALOHA, clients can transmit messages without knowing if other clients are in the process of transmitting simultaneously. There are no reservations or multiplexing techniques. The basic principle is the hub (or gateway in the case of LoRaWAN) immediately retransmits packets it has received. If an endpoint notices one of its packets was not acknowledged then it will wait then retransmit the packet. In LoRaWAN, collisions only occur if transmissions use the same channels and spreading frequency.

LoRaWAN topology

LoRaWAN is based on a star network topology. For that matter, it can be said it supports star of stars topology. Rather than a single hub-spoke model, a LoRaWAN can assume multiple hubs. A node can be associated with more than one gateway.

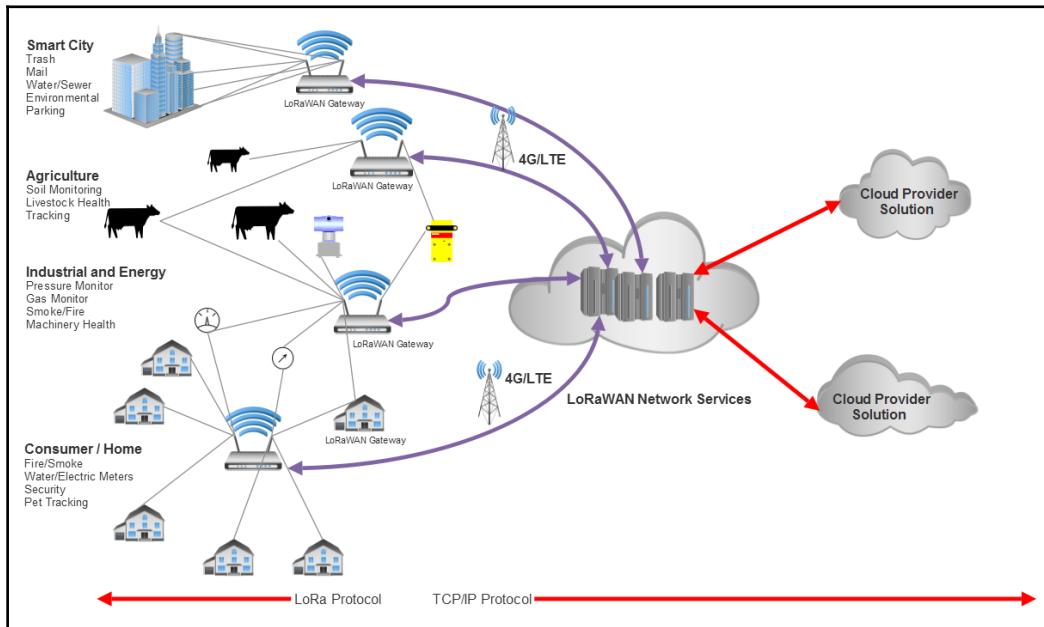


A critical component of LoRaWAN that sets it apart from most of the transports listed in this book is the fact that user data will be transported from an end-node to the gateway over the LoRaWAN protocol. At that point, the LoRaWAN gateway will forward the packet over any backhaul (such as 4G-LTE, Ethernet, Wi-Fi) to a dedicated LoRaWAN network service in the cloud. This is unique since most other WAN architectures release any control of customer data at the time it leaves their network to a destination on the internet.

The network service has the rules and logic to perform the necessary upper layers of the network stack. A side effect of this architecture is that handoff from one gateway to another isn't necessary. If a node is mobile and moves from an antenna to antenna, the network services will capture multiple identical packets from different paths. These cloud-based network services allow LoRaWAN systems to choose the best route and source of information when an end-node is associated with more than one gateway. Responsibilities of the network services include:

- Duplicate packet identification and termination
- Security services
- Routing
- Acknowledgement messages

Additionally, LPWAN systems like LoRaWAN will have 5x to 10x fewer base stations for similar coverage to a 4G network. All base stations are listening to the same frequency set, therefore they are logically one very large base station. This, of course, reinforces the statement that LPWAN systems can be a lower cost point than a traditional cellular network:



LoRaWAN network topology. LoRaWAN is built on a star of star type topology with gateways acting as the hubs as well as communication agents over traditional IP networks to LoRaWAN administration in the cloud. Multiple nodes can associate with multiple LoRaWAN gateways.

LoRaWAN summary

LoRaWAN does have gaps in the architecture and protocols that need careful consideration from the IoT architect:

- Some features common in LTE networks are simply not designed into the LoRaWAN architecture. LoRaWAN is not a complete network stack in the OSI model. It lacks the common features of a network layer, session layer, and transport layer: roaming, packetizations, retry mechanisms, QoS, and disconnects. It is up to the developer and integrator to add those services if necessary.

- LoRaWAN relies on a cloud-based network interface. At some point in the value chain, a cloud subscription is needed.
- The chip vendor is Semtech and is a single source of the technology, although there have been announced partnerships with ST Microelectronics. This is similar to the Z-Wave value chain.
- LoRaWAN is based on the ALOHA protocol. ALOHA complicates validation and acknowledgment as well as contributes to error rates over 50%.
- Downlink ability is still limited. LoRaWAN is fundamentally a one-way protocol. In certain use cases that is sufficient, but it tempers flexibility.
- LoRaWAN has high latency and no real-time ability.
- No OTA firmware updates.
- Mobility and moving nodes are challenging to manage in LoRaWAN. A 40 to 60 byte message can take 1 to 1.5 s to transmit. This can be problematic for moving vehicles at highway speed. A base station also requires **Linear Time Invariant (LTI)** which means the radio waveform shouldn't change the time of flight.
- Geo-location precision is about 100 m. Using RSSI signal strength measurements or time-of-flight measurements, a degree of accuracy can be obtained. The best solution is if three base stations triangulate a node. More base stations can improve the accuracy.

Sigfox

Sigfox is a narrowband LPWAN (like NB-IOT) protocol developed in 2009 in Toulouse, France. The founding company goes by the same name. This is another LPWAN technology using the unlicensed ISM bands for a proprietary protocol. Sigfox has some traits that significantly narrow its utility:

- Up to 140 messages per device daily on uplink (duty cycle of 1%, 6 messages/hour).
- A payload size of 12 bytes for each message (uplink) and 8 bytes (downlink).
- A throughput of up to 100 bps uplink and 600 bps downlink.

Originally, Sigfox was unidirectional and intended as a pure sensor network. That implies that only communication from the sensor uplink was supported. Since then a downlink channel has become available.

Sigfox is a patented and closed technology. While their hardware is open, however, the network is not and must be subscribed to. Sigfox hardware partners include Atmel, TI, Silicon Labs, and others. Sigfox builds and operates its network infrastructure similar to the arrangement of LTE carriers. This is a very different model than LoRaWAN. LoRaWAN requires a proprietary hardware PHY be used on their network, while Sigfox uses several hardware vendors but a single managed network infrastructure. Sigfox calculates rates by the number of devices that are connected to a customer's network subscription, the traffic profile per device, and the duration of the contract.

While there are strict limitations of Sigfox in terms of throughput and utilization, it is intended for systems sending small and infrequent bursts of data. IoT devices like alarm systems, simple power meters, and environmental sensors would be candidates. Data for various sensors can typically fit within the constraints. (such as temperature/humidity data of 2 bytes at 0.004 degree precision). One must be careful with the degree of precision the sensor provides and the amount of data that can be transmitted. One trick to use is state data; a state or event can simply be the message without any payload. In that case, it consumes 0 bytes. While this doesn't eliminate the restrictions in broadcasting, it can be used to optimize for power.



Sigfox physical layer

As mentioned previously, Sigfox is **Ultra Narrow Band (UNB)**. As the name implies, the transmission uses a very narrow channel for communication. Rather than spreading the energy across a wide channel, a narrow slice of that energy is confined to these bands:

- 868 MHz: Europe (ETSI 300-200 regulations)
- 902 MHz: North America (FCC part 15 regulations)



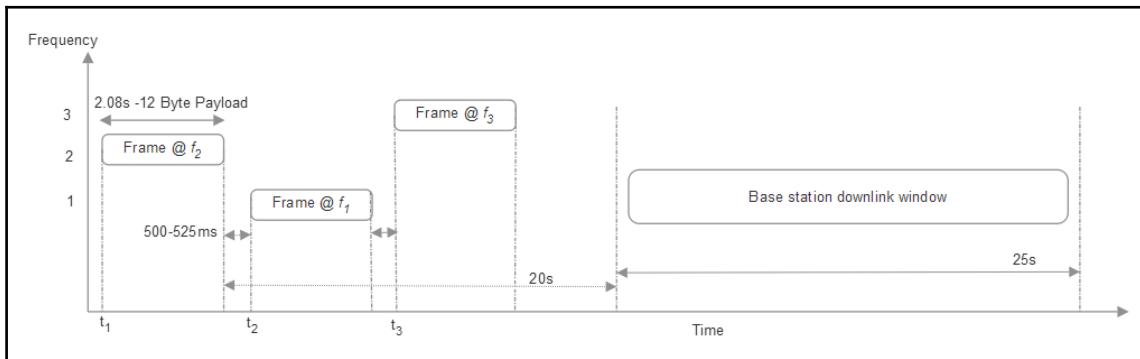
Some regions such as Japan have strict spectral density limitations currently making ultra narrowband difficult to deploy.

The band is 100 Hz in width and uses **Orthogonal Sequence Spread Spectrum (OSSS)** for the uplink signal and 600 Hz using **Gaussian Frequency Shift Keying (GFSK)** for the downlink. Sigfox will send a short packet on a random channel at a random time delay (500 to 525 ms). This type of encoding is called **Random Frequency and Time Division Multiple Access (RFTDMA)**. Sigfox, as stated, has strict parameters of use, notably data size limitations. The following table highlights these parameters for the uplink and downlink channels:

	Uplink	Downlink
Payload Limit (bytes)	12	8
Throughput (bps)	100	600
Maximum Messages per Day	140	4
Modulation Scheme	DBPSK	GFSK
Sensitivity (dBm)	<14	<27

Bidirectional communication is an important characteristic of Sigfox versus other protocols. However, bidirectional communication in Sigfox does require some explanation. There is no passive reception mode, meaning a base station cannot simply send a message to an endpoint device at any time. A receive window opens up for communication only after the transmission windows complete. The receive window will open only after a 20 second period after the first message was sent by an endpoint node. The window will remain open for 25 seconds allowing reception of a short (4 byte) message from a base station.

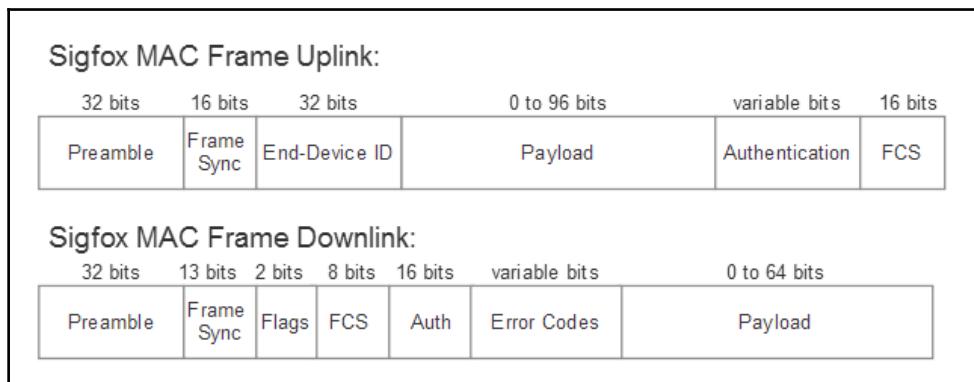
333 channels, each 100 Hz in width, are used in Sigfox. The receiver sensitivity is -120 dBm/-142 dBm. Frequency hopping is supported using a pseudo-random method of 3 out of the 333 channels. Finally, the transmission power is specified to be +14 dBm and +22 dBm in North America:



Sigfox transmission timeline. Three copies of the payload are transmitted on three unique random frequencies with various time delays. A window of downlink transmission opens only after the last uplink.

Sigfox MAC layer

Each device in a Sigfox network has a unique Sigfox ID. The ID is used for the routing and signing of messages. The ID is used to authenticate the Sigfox device. Another characteristic of Sigfox communication is that it uses *fire and forget*. Messages are not acknowledged by the receiver. Rather, a message is sent three times on three different frequencies at three different times by the node. This assists in ensuring the integrity of transmission of a message. The fire and forget model has no way of ensuring the message was actually received, so it is up to the transmitter to do as much as possible to ensure accurate transmission:



Sigfox MAC frame packet Structure for uplink and downlink

The frames contain a preamble of predefined symbols used for synchronization in transmission. The frame sync fields specify the types of frames being transmitted. The FCS is a **Frame Check Sequence (FCS)** used for error detection.

No packet contains a destination address or other node. All data will be sent by the various gateways to the Sigfox cloud service.

The data limit can be understood and modeled from the MAC layer packet format:

$$\frac{\sim 200 \text{ bit uplink packet}}{100 \text{ bps}} = 2 \text{ seconds}$$

Given each packet is transmitted three times, and we know that European regulations (ETSI) limit transmission to a 1% duty cycle, we can calculate the number of messages per hour using the maximum payload size of 12 bytes:

$$3600 \text{ seconds} @ 1\% \text{ duty cycle} = \frac{36 \text{ seconds message transmission time}}{\text{hour}} \times \frac{\text{message}}{3 \text{ repetitions} \times 2 \text{ seconds}} = \frac{6 \text{ messages}}{\text{hour}}$$

Even though twelve bytes is the limit of a payload, that message may take over one second to transmit. Early versions of Sigfox were unidirectional but the protocol now supports bi-directional communication.

Sigfox protocol stack

The protocol stack is similar to other stacks following the OSI model. There are three levels, detailed here:

- **PHY Layer:** Synthesizes and modulates signals using DBPSK in the uplink direction and GFSK in the downlink direction as previously detailed.
- **MAC Layer:** Adds fields for device identification/authentication (HMAC) and error correcting code (CRC). The Sigfox MAC does not provide any signaling. This implies that devices are not synchronized with the network.
- **Frame Layer:** Generates the radio frame from application data. Also, it systematically attaches a sequence number to the frame:

Sigfox Protocol Stack		Simplified OSI Model
Application Layer		7. Application Layer
Frame		6. Presentation Layer
MAC Layer		5. Session Layer
PHY Layer (868MHz / 902MHz Radios)		4. Transport Layer
		3. Network Layer
		2. Data Link Layer
		1. Physical Layer

Sigfox Protocol Stack versus simplified OSI model

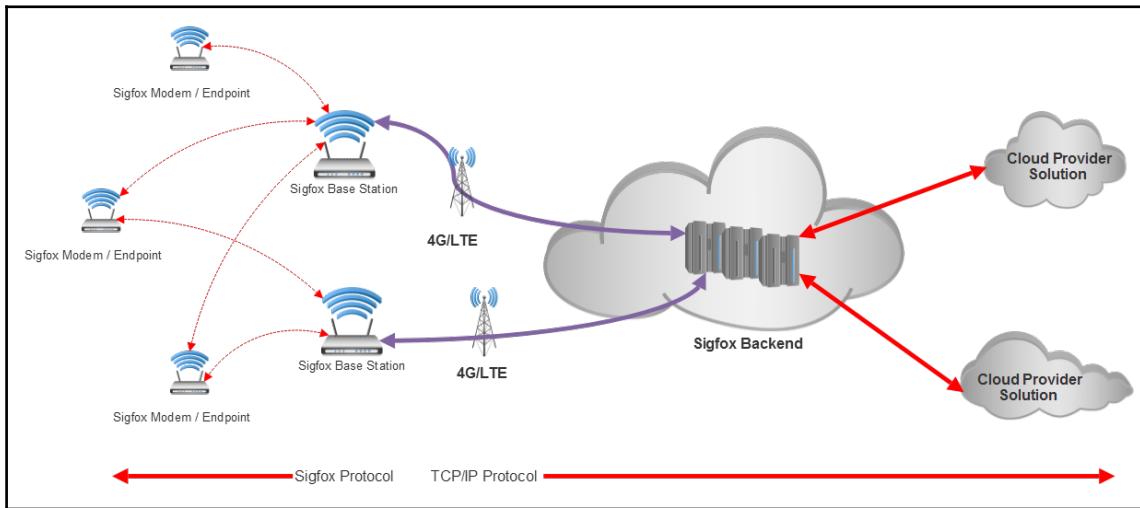
With regards to security, messages are not encrypted anywhere in the Sigfox protocol stack. It is up to the customer to provide an encryption scheme to the payload data. No keys are exchanged over a Sigfox network, however, each message is signed with a key unique to the device for identification.

Sigfox topology

A Sigfox network can be as dense as one million nodes per base station. The density is a function of the number of messages sent by the network fabric. All nodes that attach to a base station will form a star network.

All data is managed through the Sigfox backend network. All messages from a Sigfox Base Station must arrive at the backend server through IP connectivity. The Sigfox backend cloud service is the only destination for a packet. The backend will store and send a message to the client after authenticating it and verifying there are no duplicates. If data needs to be transmitted to an endpoint node, the backend server will select a gateway with the best connectivity to the endpoint and forward the message on the downlink. The backend has already identified the device by packet ID and prior provisioning will force the data to be sent to a final destination. In a Sigfox architecture, a device cannot be directly accessed. Neither the backend nor the base station will directly connect to an endpoint device.

The backend is also the administration, licensing, and provide service for customers. The Sigfox cloud routes data to the destination the customer chose. The cloud service offers APIs via a pull model to integrate Sigfox cloud functions in a third-party platform. Devices can be registered through another cloud service. Sigfox also offers callbacks to other cloud services. This is the preferred method to retrieve data:



Sigfox topology. Sigfox utilizes its proprietary non-IP protocol and converges the data to a Sigfox network backend as IP data.

To assist with ensuring data integrity of the fire and forget communication model, multiple gateways may receive a transmission from a node; all the subsequent messages will be forwarded to the Sigfox backend and duplicates will be removed. This adds a level of redundancy to the reception of data.

Attaching a Sigfox endpoint node is intended to ease installation. There is no pairing or signalization.

Summary

While there is some commonality between the types of long-range communication technologies, they also are targeting different use cases and segments. The IoT architect should choose wisely which long-range system they intend to adopt. Like other components of the IoT system, the LPWAN is difficult to change once deployed. Considerations in picking the correct LPWAN include:

- What data rate does the IoT deployment need to use?
- Can the solution scale with the same LPWAN across regions? Is there proper coverage or does it need to be built?
- What range of transmission is appropriate?
- Is there any latency requirement for this IoT solution? Can the solution work with very high (multiple seconds) latencies?
- Are the IoT endpoints battery powered and what is the cost of servicing them?
- What are the cost constraints of the endpoints?



A few other LPWAN technologies also exist such as Ingenu in the USA and the Weightless Technologies (i.e. Weight-N, Weightless-W, and Weightless-P). Weightless is interesting as it is the only LPWAN operating in the television white space. MuLTE Fire is a new technology using unlicensed frequency bands developed and promoted by Qualcomm and Nokia. It provides LTE cellular services with Wi-Fi like deployment, essentially enabling a private LTE operator to deploy a private cellular network.

For reference, the following table details the similarities and differences between the LPWAN protocols highlighted in this chapter:

Specification	Cat 0 (LTE-M) Release 12	Cat 1 Release 8	Cat M1 Release 13	Cat-NB Release 13	LoRa/LoRaWAN	Sigfox
ISM Bands	No	No	No	No	Yes	Yes
Total Bandwidth	20 MHz	20 MHz	1.4 MHz	180 kHz	125 kHz	100 kHz
Downlink Peak Rate	1 Mbps	10 Mbps	1 Mbps or 375 Kbps	200 Kbps	0.3 to 5 Kbps adaptive	100 bps

Uplink Peak Rate	1 Mbps	5 Mbps	1 Mbps or 375 Kbps	200 Kbps	5 Kbps to 5 Kbps adaptive	600 bps
Range	LTE Range	LTE Range	~4x Cat-1	~7x Cat-1	5 kM urban, 15 kM rural	up to 50 kM
Maximum Coupling Loss (MCL)	142.7 dB	142.7 dB	155.7 dB	164 dB	165 dB	168 dB
Sleep Power	Low --	High ~2mA idle	Very Low ~15 uA Idle	Very Low ~15 uA Idle	Extremely Low 1.5 uA	Extremely Low 1.5uA
Duplex Configuration	Half/Full	Full	Half/Full	Half	Half	Half
Antennas (MIMO)	1	2 MIMO	1	1	1	1
Latency	50-100 ms	50-100 ms	10-15 ms	1.6-10 s	500 ms to 2 s	up to 60 seconds
Transmit Power (UE)	23 dB	23 dB	20 dB	23 dB	14 dB	14 dB
Design Complexity	50% Cat-1	Complex	25% Cat-1	10% Cat-1	Low	Low
Cost (relative pricing)	~\$15	~\$30	~\$10	~\$5	~\$15	~\$3
Mobility	Mobile	Mobile	Mobile	Limited	Mobile	Limited

Having covered the capture of data from a sensor to the communication of that data through PAN and WAN architecture, it is now time to discuss the marshaling and processing of IoT data. The next chapter is the first treatment of that data to package it, secure it, and route it to the correct location. That location may be an edge/fog node or the cloud. We will discuss the role of the gateway in providing that networking bridge and the ability to process data at the edge. We also will detail the type of IP-based communication protocols like MQTT and CoAP needed to stream data from the edge to the cloud. Later chapters will cover the ingestion, storage, and analysis of the data generated by the IoT.

8

Routers and Gateways

IoT receives much industry and economic focus because of the number of devices that will be eventually be deployed and the amount of data these devices will produce. There are two methods regarding how IoT will take form:

- Edge-level sensors and devices will provide a direct path to the cloud. This implies that these edge-level nodes and sensors will have enough resources, hardware, software, and service-level agreements to transmit data across the WAN directly.
- Edge-level sensors will form aggregations and clusters around gateways and routers to provide staging areas, protocol conversions, and edge/fog processing abilities, and will manage security and authentication between the sensors and the WAN.

The first model is difficult and costly for power and cost-constrained edge-level sensors/actuators/devices. A more logical form will be the latter case. The role of a router or gateway at the edge of sensors/devices includes the formal networking abilities modern routers provide, such as link routing, port forwarding, tunneling, security, and provisioning. Principles of TCP/IP routing and bridging are provided in a plethora of literature and are too broad to study here. This chapter explores the role and necessity for edge-level routers and provides tips and recommendations as to the features to consider when deploying a mass IoT solution.

Routing functions

When considering IoT architecture, the router plays a significant role in the overall management, scaling, and security of a system. Often, it is the case that the role of a router is trivialized to simply act as a gateway from one protocol to another (cellular to Bluetooth converter). For a commercial or industrial deployment, significantly more needs to be considered, especially when devices are in remote and moving systems.

Gateway functions

We have covered several types of wireless communication in this book, all of which require some form of transmitter and receiver as well as a mechanism to bridge traffic to the internet. This is the single most important role of the IoT edge gateway. Whether the medium is a Bluetooth mesh requiring a bridge node or a cellular network eNodeB, the roles are similar. The gateway bridges and marshals data between two dissimilar networks.

A router can be a gateway. Routers manage and steer traffic between similar networks. In IoT architecture, one may have a 6LoWPAN mesh that is IPv6 addressable, communicating to the outside world through a device that acts as a gateway bridging 802.15.4 protocols to Wi-Fi or 802.3 physical transports, but also steering and routing data using the IP layer shared between the internet and the 6LoWPAN mesh.

Often, it is the case that the edge gateway is the central PAN controller as well. This implies that all the functionalities of managing the PAN network, securing, authenticating, and provisioning new nodes, steering data, and power-managing devices, are the responsibility of the edge gateway.

Routing

The basic function of a router is to bridge connections between network segments. Routing is considered a layer-three function of the OSI standard model because it makes use of the IP address layer to guide packet movement. Essentially, all routers rely on a routing table to guide the flow of data. The routing table is used to find the best match to a destination IP address of a packet.

There are several proven algorithms used for efficient routing. One type of routing is dynamic routing, where the algorithms react to changes in the network and topology. Information about the state of the network is shared through a routing protocol on a timed basis or upon a triggered update. Examples of dynamic routing are distance-vector routing and link-state routing. Alternatively, static routing is important and useful for small networks that need specific paths configured between routers. Static routes are non-adaptive, thus have no need to scan topologies or update metrics. These are preset on the router:

- **Shortest path routing:** A graph is constructed representing routers on the network. An arc between nodes represents a known link or connection. The algorithm simply finds the shortest path from any source to any destination.
- **Flooding:** Every packet is repeated and broadcast from every router to every endpoint on its link. This generates a massive number of duplicate packets and requires a hop counter in the packet header to ensure packets have a limited time to live. An alternative is selective flooding, which only floods a network in the general direction of the destination. Flood networks are the basis of Bluetooth mesh networking.
- **Flow-based routing:** Inspects the current flow in the network before determining a path. For any given connection, if the capacity and average flow are known, then compute the mean packet delay on that link. This algorithm finds the minimum average.
- **Distance-vector routing:** The router table contains a *best-known distance* to each destination. The tables are updated by neighboring routers. The table contains an entry for each router in the subnet. Each entry contains a preferred route/path and the estimated distance to the destination. The distance can be a metric of a number of hops, latency, or queue lengths.
- **Link-state routing:** A router initially discovers all its neighbors through a special HELLO packet. The router measures the delay to each of its neighbors by sending an ECHO packet. That topology and timing information is then shared with all routers on the subnet. A complete topology is built and shared between all routers.
- **Hierarchical routing:** Routers are divided into regions and have a hierarchical topology. Each router maintains an understanding of its own region, but not the entire subnet. Hierarchical routing is also an efficient means of controlling router table size and resources in constrained devices.

- **Broadcast routing:** Each packet carries a list of destination addresses. A broadcast router investigates the addresses and determines the set of output lines to transmit to the packet. The router will generate a new packet for each output line and include only the destinations needed in that newly formed packet.
- **Multicast routing:** The network is partitioned into well-defined groups. An application can send a packet to an entire group rather than a single destination or a broadcast.



An important metric in routing is the time of convergence. Convergence occurs when all routers in a network share the same topological information and state.

Typical edge routers will support routing protocols such as **Border Gateway Protocol (BGP)**, **Open Shortest Path First (OSPF)**, **Routing Information Protocol (RIP)**, and RIPng. An architect using edge routers in the field needs to be aware of the congestion and costs of using a certain routing protocol over others, especially if the interconnection between routers is a data-capped WAN connection:

- **BGP:** BGP-4 is the standard for internet domain routing protocols, and is described in RFC 1771; it is used by most ISPs. BGP is a distance-vector dynamic routing algorithm and advertises entire paths in routing update messages. If routing tables are large this requires substantial bandwidth. BGP will send a 19 byte keep-alive message every 60 seconds to maintain a connection. BGP may be a poor routing protocol for mesh topologies since BGP maintains a connection to neighbors. BGP also struggles with the growth of routing tables in large topologies. BGP is also unique as it's one of the only routing protocols based on TCP packets.
- **OSPF:** This protocol is described in RFC 2328 and provides network scaling and convergence advantages. Internet backbone and enterprise networks use OSPF heavily. OSPF is a link-state algorithm that supports IPv4 and IPv6 (RFC 5340) and operates on IP packets. It has the advantage of detecting dynamic link changes in seconds and responding.

- **RIP:** Version two of RIP is a distance-vector routing algorithm based on hop counts using an interior gateway protocol. Originally based on the Bellman-Ford algorithm, it now supports variable-size subnets, overcoming the limitations in the original version. Loops in a routing table are restricted by limiting the maximum number of hops in a path (15). RIP is UDP-based and supports only IPv4 traffic. RIP has a longer convergence time than protocols such as OSPF but is easy to administer for small edge router topologies. Still, convergence for RIP with only a few routers can take several minutes.
- **RIPng:** RIPng stands for RIP next generation (RFC 2080). This allows for the support of IPv6 traffic and IPsec for authentication.

A typical routing table used in a product like the Cradlepoint IBR900 router is as follows:

```
[administrator@IBR900-e11: /]$ route
Table: wan
Destination      Gateway     Device  UID      Flags   Metric  Type
default          96.19.152.1 wan        onlink    0       unicast

Table: main
Destination      Gateway     Device  UID      Flags   Metric  Type
96.19.152.0/21   *           wan        0       unicast
172.86.160.0/20  *           iface:pertino0 0       unicast
172.86.160.0/20  None        None      256     blackhole
192.168.1.0/24   *           primarylan 0       unicast
2001:470:813b::/48 *           *iface:pertino0 256     unicast
fe80::/64         *           lan       256     unicast

Table: local
Destination      Gateway     Device  UID      Flags   Metric  Type
96.19.152.0      *           wan        0       broadcast
96.19.153.13    *           wan        0       local
96.19.159.255   *           wan        0       broadcast
127.0.0.0         *           *iface:lo  0       broadcast
.
.
.
```

In this example, there are three tables: `wan`, `main`, and `local`. Each table contains particular routing paths particular to that interface:

- **Destination:** The complete or partial IP address of the packet destination. If the table contains the IP, it will reference the rest of the entry to resolve the interface to route to a partial address can be given prefixed by `/`. This specifies the fix bit positions of the address to resolve. For example, `/24` in `192.168.1.0/24` specifies the top 24 bits of `192.168.1` are fixed and the lower 8 bits can resolve any address on the `192.168.1.*` subnet.
- **Gateway:** This is the interface to direct packets to that match the destination lookup. In the previous case, the gateway is specified as `96.19.152.1` and the destination is `default`. This implies the outbound WAN at `96.19.152.1` will be used for all destination addresses. This is essentially an IP passthrough.
- **Device UID:** This is an alphanumeric identifier for the interface to direct data to. For example, any destination in the `172.19.152.0/21` subnet will route packets to an interface labeled `iface:pertino0`. Often, this field will be expressed with a numeric IP address rather than a symbolic reference.
- **Flags:** Used for diagnostics and indicate route state. States could be `route up`, `use gateway`.
- **Metric:** This is the distance to the destination and is usually counted by the number of hops.
- **Type:** Several route types can be used:
 - `unicast`: The route is a real path to a destination.
 - `unreachable`: Destination is unreachable. Packets are discarded and an ICMP message indicating the host is unreachable is generated. The local sender receives an `EHOSTUNREACH` error.
 - `blackhole`: Destinations are unreachable. Unlike prohibit types, packets will drop quietly. Local senders get an `EINVAL` error.
 - `prohibit`: Destinations are unreachable. Packets will be discarded and ICMP messages generated. Local senders receive an `EACCES` error.
 - `local`: Destinations are assigned to this host. The packets are looped back and delivered locally.
 - `broadcast`: Packets will be sent as broadcasts over the interface to all destinations.

- **throw:** Special control route to force packets to drop and generate ICMP unreachable messages.

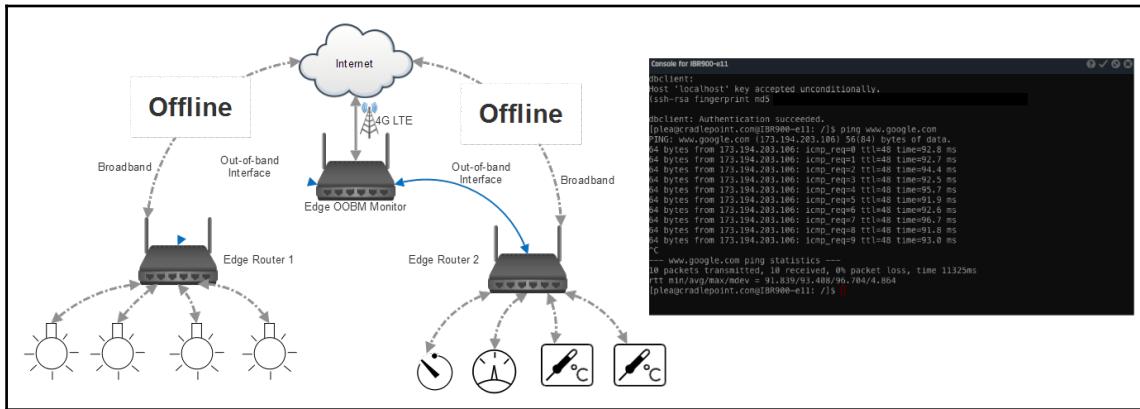
It should also be noted in the preceding example that IPv6 addresses are intermixed with IPv4 addresses. For example, `2001:470:813b::/48` in the main table is an IPv6 address with a /48 bit subnet.

Failover and out-of-band management

Failover is a critical feature for certain IoT edge routers, particularly for moving vehicles and patient care applications. Failover, as the name implies, is switching from one WAN interface to another when a primary source is lost. Loss of WAN could be attributed to the loss of cellular connectivity in a tunnel. A logistics company with a fleet of cold storage trucks may need to guarantee connectivity where cellular service varies nationwide. A failover from one cellular provider to another by use of multiple SIM identities can help mitigate and smooth over connectivity. Another use case could be using customer Wi-Fi as a primary WAN interface for in-home health monitoring, but enabling a failover to cellular WAN if the Wi-Fi signal is lost. Failover should be seamless and automatic without packet loss or a noticeable impact on data latency.

Out-of-Band Management (OOBM) should also be considered for IoT devices. OOBM is useful in fail-safe conditions where a dedicated and isolated channel is needed for the management of equipment. Sometimes this is called **light-out management (LOM)** if the primary systems are offline, damaged, or have power loss; one is still able to manage and inspect the equipment remotely through a sideband channel. In IoT, this can be useful for situations that need guaranteed uptime and remote management, such as oil and gas monitoring or industrial automation. A well-designed OOBM system should have no reliance on the system being monitored to be functional. Typical management planes such as VNC or SSH tunnels, require the device to be booted and functional.

An OOBM needs to be adjunct and isolated from the system, as shown in the following figure:

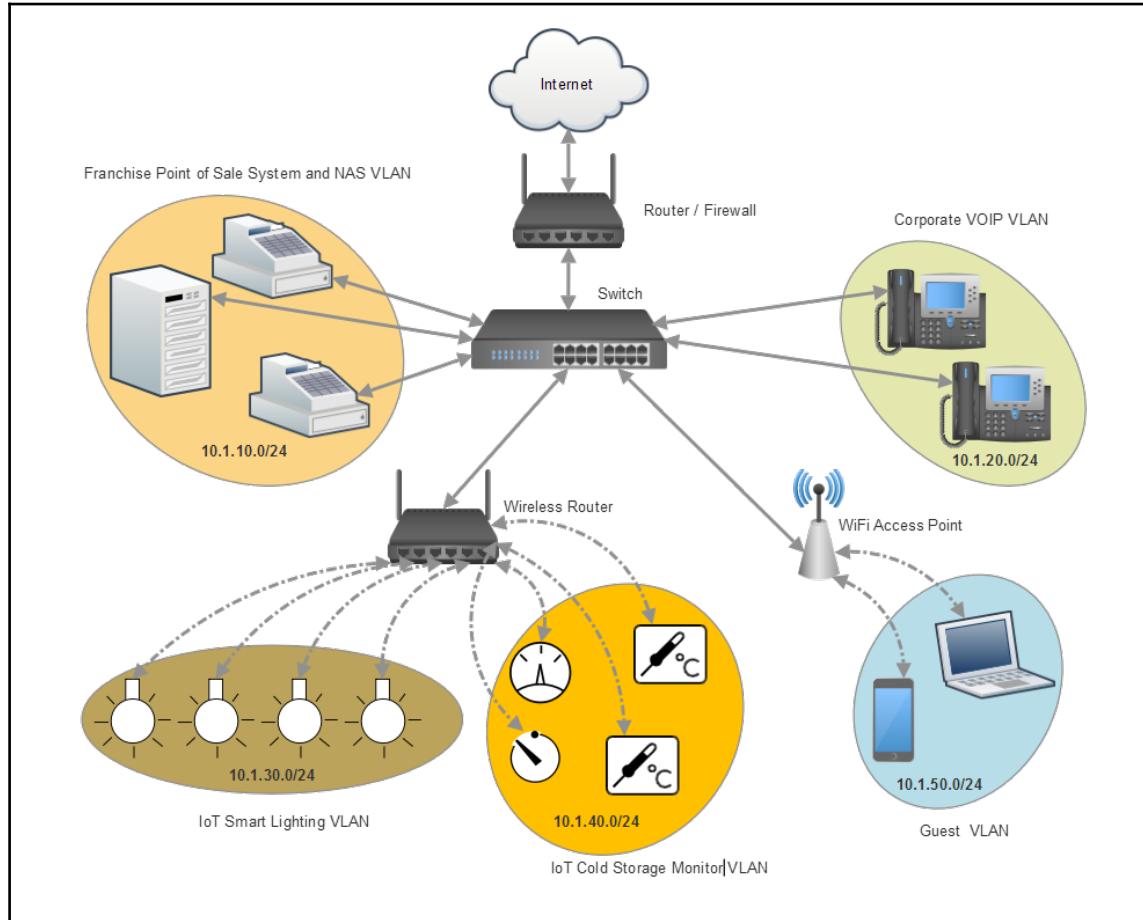


Example configuration for out-of-band management.

VLAN

A VLAN functions like any other physical LAN, but it enables computers and other devices to be grouped together even if they are not physically attached to the same network switch. The partitioning occurs in the data link layer (two) of the OSI model. VLANs are a form of network segmentation of devices, applications, or users, although they share the same physical network. A VLAN can also group hosts together, although they are not on the same network switch, essentially easing the burden of partitioning networking without running extra cables. IEEE 802.1Q is the standard by which VLANs are built. Essentially, a VLAN uses an identifier or tag consisting of 12 bits in the Ethernet frame. Therefore, there is a hard limit of 4096 potential VLANs on a single physical network.

A switch can assign a port to directly map to a particular VLAN. Since the VLAN is designed in layer two of the stack, traffic can be tunneled through layer three, allowing for geographically separated VLANs to share a common topology:



An example VLAN architecture in a franchise or retail scenario.

Shown above is a corporate point-of-sale (POS) and a VOIP system that is virtually isolated from a set of IoT devices as well as the guest Wi-Fi. This is done through VLAN addressing although the system shares the same physical network. Here we assume this is a smart IoT deployment where all edge IoT devices and sensors carry an IP stack and are addressable through the LAN.



VLAN design is particularly useful in the IoT space. Isolating IoT devices from other corporate functions is a typical scenario. VLANs are only useful when dealing with IP addressable devices.

VPN

VPN tunnels are used to establish a secure connection to a remote network over a public network. For example, VPN tunnels can be used across the internet by an individual to connect to a secure corporate network while traveling, or by two office networks to function as one network. The two networks set up a secure connection to the (normally) unsecured internet by assigning VPN encryption protocols.



For IoT deployment, a VPN is necessary to move data from remote sensors and edge devices into a corporate or private LAN. Typically, a corporation will be behind a firewall and a VPN is the only means to move data directly into a private on-premises server. In these cases, a VPN may be a necessary component of the router to bridge the networks. Later in this chapter, a discussion of software-defined networks will present an alternative method to securing networks.

Several VPNs exist:

- **Internet Protocol Security (IPSec) VPN:** The traditional form of VPN technology that resides on the network layer of the OSI stack and secures data through a tunnel between two endpoints.
- **OpenVPN:** This is an open source VPN for secure point-to-point and site-to-site connections in routed or bridged configurations. It incorporates a custom security protocol utilizing SSL/TLS (OpenSSL) for key exchanging and encrypting control and data planes. It can run over UDP and TCP transports. SSL is common in most browser applications, therefore, SSL VPN systems can provide secure tunnels on an application basis rather than an entire network.

- **Generic Routing Encapsulation (GRE):** Creates point-to-point connections between endpoints through a tunnel similar to a VPN tunnel but encapsulates its payload. It wraps this inner packet within an outer packet. This allows the data payload to pass across other IP routers and tunnels unperturbed. Additionally, GRE tunnels can transport IPv6 and multicast transmissions.
- **Layer 2 Tunneling Protocol (L2TP):** Creates a connection between two private networks via UDP datagrams typically used for VPNs or as part of the delivery services by ISPs. There is no security or encryption built into the protocol and it often relies on IPsec for that ability.

A VPN must either trust the underlying network protocols or provide for its own security. VPN tunnels typically use IPsec to authenticate and encrypt packets exchanged across the tunnels. To set up a VPN tunnel router on one end, there must be another device (usually a router) that also supports IPsec on the other end. **Internet Key Exchange (IKE)** is the security protocol in IPsec. IKE has two phases. The first phase is responsible for establishing a secure communication channel, and in the second phase, the established channel is used by IKE peers. The router has several different security protocol options for each phase, but the default selections will be sufficient for most users. Each IKE exchange uses one encryption algorithm, one hash function, and one DH group to make a secure exchange:

- **Encryption:** Used to encrypt messages sent and received by IPsec. Typical encryption standards and algorithms include AES 128, AES 256, DES, and 3DES.
- **Hash:** Used to compare, authenticate, and validate data across the VPN, ensuring it arrives in its intended form, and to derive keys used by IPsec. Typical hash functions one should expect in an enterprise-grade router include MD5, SHA1, SHA2 256, SHA2 384, and SHA2 512. Note that some encryption/hash combinations (such as 3DES with SHA2 384/512) are computationally expensive, impacting WAN performance. AES provides good encryption and performs much better than 3DES.
- **DH groups:** The DH (Diffie-Hellman) group is a property of IKE and is used to determine the length of prime numbers associated with key generation. The strength of the key generated is partially determined by the strength of the DH group. Group 5, for instance, has greater strength than group 2:
 - **Group 1:** 768-bit key
 - **Group 2:** 1024-bit key
 - **Group 5:** 1536-bit key

In IKE phase one, you can only select one DH group if you are using an aggressive exchange mode.

The algorithms are listed in order of priority. You can reorder this priority list by clicking and dragging algorithms up or down. Any selected algorithm may be used for IKE, but the algorithms at the top of the list are more likely to be used more often.



A word of caution on mobile and power-constrained IoT deployments. A traditional VPN cannot withstand moving in and out of a persistent network connection (such as cellular roaming, carrier switching, or occasionally-powered devices). If the network tunnel is disrupted, it causes timeouts, disconnects, and failures. Some mobile VPN software, such as **Host Identity Protocol (HIP)** from the IETF, attempts to resolve the problem by disassociating different IP addresses used when roaming to the VPN logical connection. Another alternative is **Software-Defined Networking (SDN)**, which will be covered later in this book.

Traffic shaping and QoS

Traffic shaping and **quality of services (QoS)** features are useful in deployments that need a guaranteed level of service when dealing with congestion or variable network loads. For example, in an IoT use case, when mixing live video streams as well as public Wi-Fi, the video feeds may need priority and a guaranteed level of quality, especially in a situation of public safety or surveillance. Left to itself, incoming data from the WAN to an edge router will be serviced on a first-come-first-serve basis:

- **QoS functions:** Allow the administrator to assign levels of priority for a given IP address hosted by the router or a specific port. QoS functions only control the uplink channel. They are particularly useful in cases where the uplink channel has much less capacity than the downlink. Typically, consumer broadband will have something like a 5 Mbps uplink and a 100 Mbps downlink, and QoS does provide a method of load balancing the constrained upstream link. QoS does not assign hard limits nor segment the link like traffic shaping does.
- **Traffic shaping functions:** Traffic shaping is a static form of pre-allocating bandwidth. For example, a 15 Mbps link could be partitioned into smaller 5 Mbps segments. These segments would be pre-assigned. Generally, this is wasteful since that bandwidth would not necessarily be returned back to the aggregate if needed.

- **Dynamic shaping and packet priority:** Modern routers enable dynamic shaping attributes. These allow the administrator to dynamically assign bandwidth segmentation rules to both ingress and egress traffic. It also can manage latency sensitive packets (such as video or user interface) for real-time applications. Dynamic shaping and packet priority allow for rules to be created based on the type of data or application rather than just the IP address or port.



A method to classify and manage network traffic is **Differentiated Services (DiffServ)**. DiffServ uses a 6-bit differentiated service code point (DSCP) in the IP header for packet classification. The concept of DiffServ is that complex functions (such as packet classification and policing) can be carried out at the edge of the network by edge routers which then mark the packet to receive a particular type of per-hop behavior. Traffic entering a DiffServ router is subject to classification and conditioning. Additionally, a DiffServ router is free to change the classification of a previously marked packet by a different router. DiffServ is a coarse-grain tool for traffic management since the chain of routers in a link do not all need to support it. The router will then manage the different packet classes through QoS features. Alternatively, IntServ, which stands for *Integrated Services*, assists in QoS and mandates all routers in the chain support it. This is a form of fine-grain QoS.

Another aspect of network quality is the **Mean Opinion Score (MOS)**. MOS is the arithmetic mean of individual values on a scale of the quality of a system from a user perspective. This is commonly used in **Voice over Internet Protocol (VOIP)** applications but certainly can be used for vision systems, imaging, streaming data, and user interface usability. It is based on a subjective rating of one to five (one being the worst quality, five being the best quality), and should be used in a feedback loop to increase capacity or reduce data sizes to match capacity.

An edge router bridging a PAN to an IP-based WAN has several options at its disposal to respond to changes in link quality and degradation of network services, for example, in an IoT deployment for fleet trucking where the carrier signal may degrade. In these situations, the router can employ **TCP Performance Enhancing Proxies (PEP)** to overcome and compensate for quality changes (RFC 3135). PEPs can be used in the transport layer or application layer of the stack and differ based on the physical medium. Forms of PEP include:

- **Proxy PEP:** Here, the proxy acts as the middle man to mimic an endpoint
- **Distribution PEP:** A PEP can run on one end of the link or both (distribution model)

A PEP consists of the following functions:



- **Split TCP:** The PEP breaks the end-to-end connection into multiple segments to overcome large delay times that affect TCP windows. These are typically used in satellite communication.
- **ACK filtering:** In links where the data rates are not symmetric (such as Cat-1: 10 Mbps down, 5 Mbps up), an ACK filter helps by accumulating or decimating TCP ACKs to improve performance.
- **Snooping:** This is a form of integrated proxy and is used to hide interference and collisions on wireless links. It intercepts duplicate ACKs in the network and drops and replaces them with a lost packet. This prevents the sender from arbitrarily reducing the TCP window size.
- **D-Proxy:** PEP to assist in wireless networks by distributing a TCP proxy to each side of the link. The proxies monitor TCP data packet sequence numbers by looking for packet loss. Upon detection, the proxies open a temporary buffer and absorb packets until the missing one is recovered and re-sequenced.

Security functions

An edge router or gateway fulfills another important role by providing a layer of security between the WAN, internet, and the underlying PAN/IoT devices. Many devices will lack the necessary resources, memory, and computational power to provide robust security and provisioning. Whether the architect is building their own gateway service or purchasing one, the following list of features should be considered to secure IoT components.

Firewall protection is the most basic form of security. There are two basic forms of firewalls for telecommunications. The first is a network firewall which filters and controls information flow from one network to another. The second is a host-based firewall that protects applications and services local to that machine. In the case of IoT edge routers, we focus on network firewalls. By default, a firewall will prevent certain types of network traffic from flowing into a firewall-protected zone, but any traffic originating within that zone is allowed to flow outward. A firewall will find and isolate information based on packets, states, or applications depending on the sophistication of the firewall. Typically, zones will be created around network interfaces with rules designed to control traffic flow between zones. An example would be an edge-router with a guest Wi-Fi zone and a corporate private zone.

A packet firewall can isolate and contain certain traffic based on the source or destination IP, ports, MAC addresses, IP protocols, and other information contained in the packet header. A stateful firewall operates in layer four of the OSI stack. It gathers and aggregates packets looking for patterns and state information such as new connections versus existing connections. Application filtering is yet more sophisticated in that it can search for certain types of application network flows, including FTP traffic or HTTP data.

A firewall can also utilize a **demilitarized zone (DMZ)**. A DMZ is simply a logical zone. A DMZ host is effectively not firewalled, in the sense that any computer on the internet may attempt to remotely access network services at the DMZ IP address. Typical uses involve running a public web server and sharing files. The DMZ host is usually specified by a direct IP address.

Port forwarding is a concept that enables certain ports behind a firewall to be exposed. Several IoT devices need an open port to provide services that are controlled by cloud components. Again, a rule is constructed that allows a specified IP address within the firewalled zone to have an exposed port.



Both DMZs and port forwarding open ports and interfaces within an otherwise protected network. Caution should be exercised to ensure this is the intention of the architect. In a mass IoT deployment with many edge routers, a generic rule to open a port may be useful for one location but a significant security risk for another. Additionally, keeping an audit of DMZ and open ports should be a security process as network topology and configurations change with time. A DMZ, at any one time, could lead to an open hole in the network security later.

Metrics and analytics

In many instances, IoT edge devices will be under service-level agreements or data caps when using metered data such as 4G LTE. In other situations, the edge router or gateway is a host of other PAN networks and IoT devices. It should serve as a central (but local) authority on the health of the PAN network/mesh. Metrics and analytics are useful for collecting and resolving connectivity and cost challenges, especially as the scale of IoT grows. Typical metrics and collections should include the following:

- **WAN uptime analytics:** Historical trends, service levels
- **Data usage:** Ingress, egress, aggregate, per client, per application
- **Bandwidth:** Random or scheduled bandwidth analysis on ingress and egress
- **PAN health:** Bandwidth, abnormal traffic, mesh reorganization
- **Signal integrity:** Signal strength, site survey
- **Location:** GPS coordinates, movement, location changes
- **Access control:** Clients attached, administrator logins, router configuration changes, PAN authentication success/failure
- **Failover:** Number of failover events, time and duration

These types of metrics should be collected and monitored on a scheduled and automatic basis. Additionally, advanced routers should be able to construct rules and alerts when certain events or abnormal behaviors are seen at the edge.

Edge processing

As we will explore later in the chapter regarding cloud and fog computing, edge routers are capable of providing computing resources close to where the data is being generated. This is an important characteristic in edge routers, especially for IoT solutions. As more IoT solutions are built in remote and varied locations, having the ability to compute locally is essential for many use cases. Computing at the edge can involve affecting user data through techniques such as:

- Filtering and aggregation
- Denaturing data
- Security and intrusion detection analysis
- Key management
- Rules engines/event processors
- Caching and storage

Edge routers will vary in size and computational capacity. They are not data center servers or rackmount hardware. Typically, the resources on an edge router will be comparable to the following table:

Feature	Consumer-grade router	Mid-range edge router	High-end fog node
Brand	Apple Airport Extreme	Advantech WISE-3310	HP Edgeline EL20 Gateway
MSRP	~\$199	~\$547	~\$1400
SOC/processor Mfr.	Broadcom 53019	Freescale i.MX6	Intel 4300U
CPU type and speed	2-core ARM A9 @ 1 GHz	ARM A9 @1 GHz	Dual-core Intel Core i5 @1.9GHz
RAM	512 MB DDR3	1 GB DDR3	8 GB DDR3
Storage	32 MB serial flash	4 GB eMMC	64 GB SSD (optional SATA)

The available resources need to be shared with the primary functions of the router as a network agent as well as with whatever the user designs as edge or fog software.

An edge device needs to manage and secure customer-deployed solutions on the edge, as in many situations the device will be very far from the programmer or administrator. The edge needs to be thought of in the same context as cloud-developed software and services. A router that offers edge computing services will provide an API or an SDK interface to use resources on the device and deploy programs.

Software-Defined Networking

Software-Defined Networking (SDN) is a method of decoupling the software and algorithms that define the networking control plane from the underlying hardware that manages the forwarding plane. Additionally, **Network Function Virtualization (NFV)** is defined as providing network functions that run on vendor-agnostic hardware. NFV describes virtualizing network features typically found in layers four through seven of the stack. These two paradigms provide the industry methods to build, scale, and deploy significantly complex network architecture in a very flexible manner. Preceding all, this greatly reduces enterprise cost in network infrastructure since most of the services can run in the cloud.

Why is this important for devices at the edge and where does it fit in with the IoT? We have spent a good deal of this book detailing data movement from a sensor to a cloud, yet have taken for granted how the overall inter-networking infrastructure will scale to a billion additional nodes on a network. One needs only to ask an IT administrator to put an additional million endpoints on a corporate network where the nodes are heterogeneous, in remote locations, and some moving in vehicles to understand the impact on the overall networking infrastructure. Traditional networking does not scale, and we are forced to consider alternative means for building large networks with minimal impact and cost.

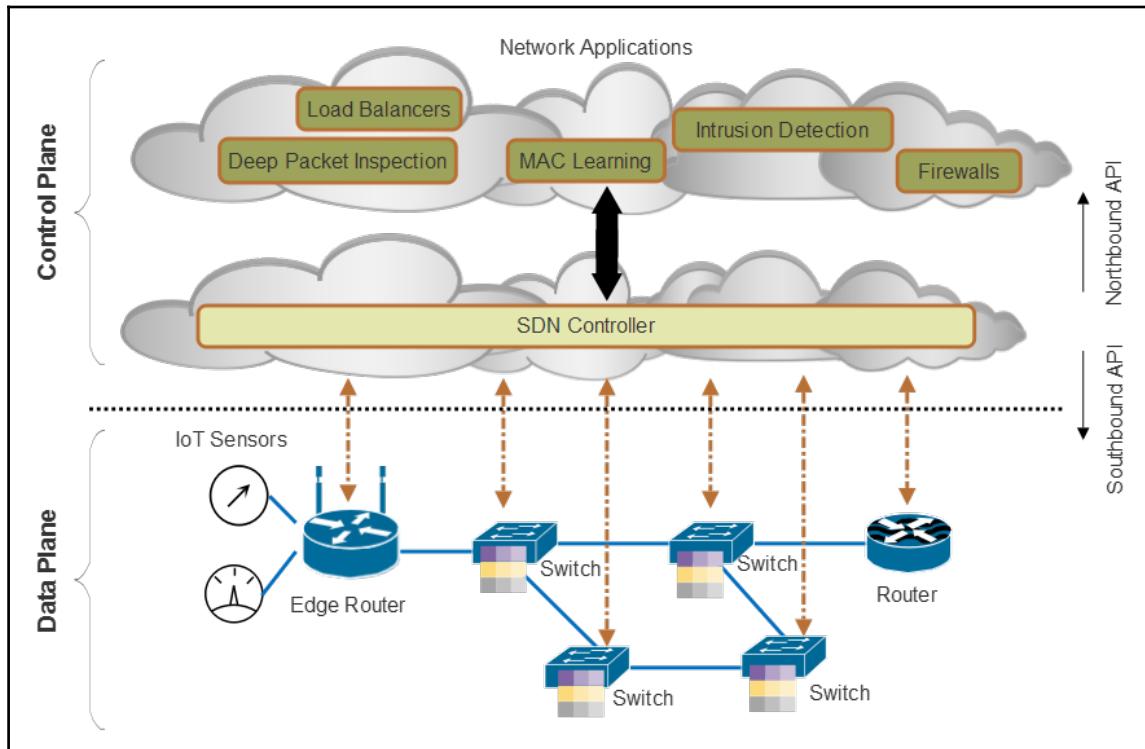
SDN architecture

The journal article *Software-Defined Networking: A Comprehensive Survey* (D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015) defines SDN as having four characteristics:

- The control plane is decoupled from the data plane. Data plane hardware becomes simple packet-forwarding devices.
- All forwarding decisions are flow-based rather than destination-based. A flow is a set of packets that match a criterion or filter. All packets in a flow are treated with the same forwarding and service policies. Flow programming allows for easy scaling and flexibility with virtual switches, firewalls, and middleware.
- The control logic is also known as the SDN Controller. This software version of legacy hardware is capable of running on commodity hardware and cloud-based instances. Its purpose is to command and govern the simplified switching nodes. The reach from the SDN controller abstraction to the switching nodes is the southbound interface.
- Network application software can reside over the SDN controller through a northbound interface. This software can interact and manipulate the data plane with services such as deep packet inspection, firewalls, and load balancers.

An SDN's infrastructure is similar to a traditional network since it utilizes similar hardware: switches, routers, and middleboxes. The principal difference, however, is that an SDN utilizes the fast server-class *off-the-shelf* computing power without complex and unique embedded control hardware. These server platforms are usually in the cloud performing network services in software rather than custom ASICs. The edge routers are essentially dumb without autonomous control. The SDN architecture breaks the control plane (the logic and function control) and the data plane (that executes datapath decisions and forwards traffic). The data plane consists of routers and switches that have an association with an SDN controller.

Everything above the data plane forwarding hardware typically can reside in the cloud or on private data center hardware:



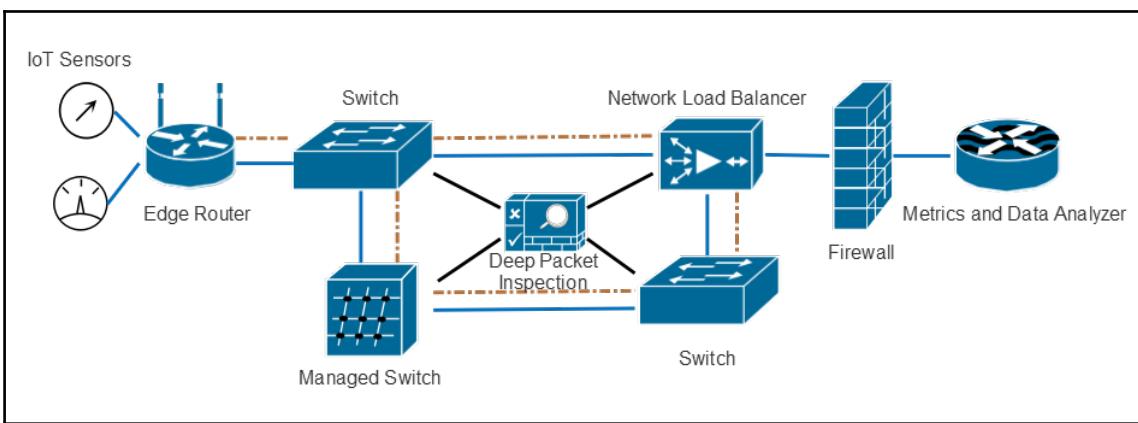
Typical SDN architecture abstraction.

The illustration shows simplified switching and forwarding nodes that reside on the data plane and marshal information along prescribed paths determined by an abstracted SDN Controller that can live in a cloud instance. The SDN Controller manages the control plane via a southbound interface to the forwarding nodes. Network applications can reside on top of the SDN Controller and manipulate the data plane with services such as threat monitoring and intrusion detection. These services typically require custom and unique hardware solutions to be deployed and managed by the customer.

Traditional internetworking

A typical internetworking architecture will use a collection of managed hardware/software components that are single-purposed and contain embedded software/solutions. Often, these use no-commodity hardware and dedicated ASIC designs. Typical functions include routing, managed switches, firewalls, deep packet inspection and intrusion detection, load balancers, and data analyzers. Often, such dedicated appliances need to be managed by the customer and staffed with trained network IT personnel to maintain and administer them. These components may come from multiple vendors and require significantly different methods of management.

In this configuration, the data plane and control plane are unified. When the system needs to add or remove another node or set up a new data path, many of the dedicated systems need to be updated with new VLAN settings, QoS parameters, access control lists, static routes, and firewall pass-throughs. This may be manageable when dealing with several thousand endpoints. However, when we scale to millions of nodes that are remote, moving and connecting/disconnecting such traditional technology regularly becomes untenable:



Traditional networking components. In a typical inter-networking scenarios, systems providing services such as security, deep packet inspection, load balancing, and metric gathering require custom vendor hardware and management systems. This introduces difficulty in management and scaling for large installations, remote devices, and moving systems since the control and data planes are unified.

SDN benefits

An SDN model of networking should be considered for mass IoT deployments, especially when a customer needs to establish the provenance and security of a wide deployment of nodes. An architect should consider the following situations when using an SDN:

- Servers and data centers that IoT edge devices must communicate with can be thousands of miles away
- The scale of IoT growth from millions of endpoints to billions of endpoints needs appropriate scaling technologies outside of the hub-and-spoke model of current internet infrastructure

Three aspects of SDN makes it attractive to IoT deployments:

- **Service chaining:** This allows a customer or provider to sell services a la carte. Cloud network services such as firewalls, deep packet inspection, VPNs, authentication services, and policy brokers can be linked and used on a subscription basis. Some customers may want a full set of features, others may not choose any or may change their configuration routinely. Service chaining allows for significant flexibility in deployments.
- **Dynamic load management:** An SDN enjoys the flexibility of cloud architecture, and by design it can scale resources dynamically depending on load. This type of flexibility is crucial for the IoT as architects need to plan for capacity and scale as the number of things grows exponentially. Only virtual networking in the cloud provides the ability to scale capacity when needed. An example of this would be people-tracking at amusement parks and other venues. The number of people varies depending on the season, time of day, and weather. A dynamic network can adjust to the number of visitors without any change to the provider's hardware.
- **Bandwidth calendaring:** This allows an operator to partition data bandwidth and usage to specified times and days. This is pertinent to IoT as many edge sensors only report data periodically or at a certain time of day. Sophisticated bandwidth sharing algorithms can be constructed to time slice capacity.

Later in this book, the Chapter 12, *IoT Security*, will explore **Software-Defined Perimeters (SDP)** as another example of network function virtualization and how it can be used to create micro-segments and device isolation, which is critical for IoT security.

Summary

Routers and gateway play an integral role in IoT development. The functionality provided by an edge router allows for enterprise-level of security, routing, resiliency, and quality of service. The gateway plays an important role in translating non-IP networks into the IP-based protocols necessary for internet and cloud connectivity. It is also important to realize that the growth of IoT to billions of nodes will take shape with low cost and constrained electronics. Features such as enterprise routing, tunneling, and VPNs require significant hardware and processing capabilities, and it makes sense to use routers and gateways for that service function. Later in this book, we will also look at how edge routers play a vital role in edge processing and fog computing.

The next chapter will go deep into IoT-based protocols such as MQTT and CoAP, with working examples. These protocols are the lightweight language of IoT and more often than not use gateways and edge routers as translators.

9

IoT Edge to Cloud Protocols

So far in this book, we have generated data or an event from a device running at the network edge. We have discussed various telecommunication mediums and technologies to move that data from the WPAN, WLAN, and WAN. There are many complexities and subtleties in building and bridging these network connections from non-IP-based PAN networks to IP-based WAN networks. There are also protocol conversions that need to be understood. Standard protocols are the tools that bind and encapsulate raw data from a sensor to something meaningful and formatted for the cloud to accept. One of the principal differences between an IOT system and M2M system is that M2M may communicate over a WAN to a dedicated server or system with no encapsulated protocol whatsoever. The IoT by definition is based on communication between endpoints and services with the internet being the common network fabric.

This chapter will also detail the necessary protocols prevalent in the IOT space such as **Message Queue Telemetry Transport (MQTT)** and CoAP.

Protocols

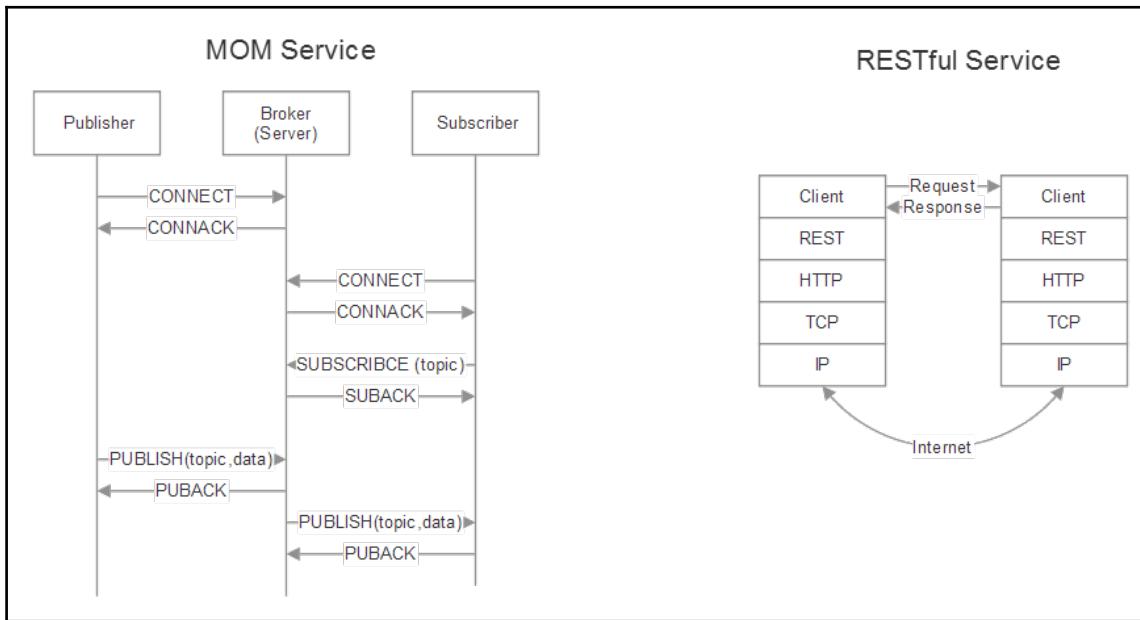
A natural question is, why are there any protocols outside of HTTP to transport data across the WAN? HTTP has provided significant services and abilities for the internet for over 20 years, yet was designed and architected for general purpose computing in client/server models. IoT devices can be very constrained, remote, and bandwidth limited. Therefore, more efficient, secure, and scalable protocols are necessary to manage a plethora of devices in various network topologies such as mesh networks.

In transporting data to the internet, designs are relegated to the TCP/IP foundation layers. TCP and UDP protocols are the obvious and only choice in data communication with TCP being significantly more complex in its implementation than UDP (being a multicast protocol). UDP, however, does not have the stability and reliability of TCP, forcing some designs to compensate by adding resiliency in the application layers above UDP.

Many of the protocols listed in this chapter are **Message Oriented Middleware (MOM)** implementations. The basic idea of a MOM is that communication between two devices takes places using distributed message queues. A MOM delivers messages from one user-space application to another. Some devices produce data to be added to a queue while others consume data stored in a queue. Some implementations require a broker or middleman be the central service. In that case, producers and consumers have a publish and subscribe-type relationship with the broker. AMQP, MQTT, and STOMP are MOM implementations; others include CORBA and Java messaging services. A MOM implementation using queues can use them for resilience in design. Data can persist in the queues, even if the server fails.

The alternative to MOM implementation is RESTful. In a RESTful model, a server owns the state of a resource but the state is not transferred in a message to the server from the client. RESTful designs use HTTP methods such as GET, PUT, POST, and DELETE to place requests upon a resource's **Universal Resource Identifier (URI)** (see the following figure). No broker or middle agent is needed in this architecture. As they are based on the HTTP stack, they enjoy the majority of services offered, such as HTTPS security. RESTful designs are typical of client-server architectures. Clients initiate access to resources through synchronous request-response patterns.

Additionally, clients are responsible for errors, even if the server fails. The figure below illustrates a MOM versus a RESTful service. On the left is a messaging service (based on MQTT) using a middle broker server and publishers and subscribers to an event. Here many clients can be both publishers and subscribers and information may or may not be stored in queues for resiliency. On the right is a RESTful design where the architecture is built upon HTTP and uses HTTP paradigms for communicating from the client to the server:



An example comparing a MOM to a RESTful implementation.

The URI is used as an identifier for web-based data traffic. The most notable URI is the **Universal Resource Locator (URL)**, such as `http://www.iotforarchitects.net:8080/iot/?id="temperature"`. The URI can be broken into component parts that are used by various levels of the network stack:



- **Scheme:** `http://`
- **Authority:** `www.iotforarchitects.net`
- **Port:** `8080`
- **Path:** `/iot`
- **Query:** `?id="temperature"`

MQTT

The IBM Websphere Message Queue technology was first conceived in 1993 to address problems in independent and non-concurrent distributed systems to securely communicate. A derivative of the WebSphere Message Queue was authored by Andy Stanford-Clark and Arlen Nipper at IBM in 1999 to address the particular constraints of connecting remote oil and gas pipelines over a satellite connection. That protocol became known as the MQTT. The goals of this IP-based transport protocol are:

- It must be simple to implement
- To provide a form of quality of service
- To be very lightweight and bandwidth efficient
- To be data agnostic
- To have continuous session awareness
- To address security issues

MQTT provides for these requirements. A way to think of the protocol is best defined by the standard body (mqtt.org) which presents a very well-defined summary of the protocol:

"MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium."

MQTT was an internal and proprietary protocol for IBM for many years until being released in version 3.1 in 2010 as a royalty-free product. In 2013, MQTT was standardized and accepted into the OASIS consortium. In 2014, OASIS released it publicly as version MQTT 3.1.1. MQTT is also an ISO standard (ISO/IEC PRF 20922).

MQTT publish-subscribe

While client-server architectures have been the mainstay for data center services for years, publish-subscribe models represent an alternative that is useful for IoT uses. Publish-subscribe, also known as **pub/sub**, is a way to decouple a client transmitting a message from another client receiving a message. Unlike a traditional client-server model, the clients are not aware of any physical identifiers such as the IP address or port. MQTT is a pub/sub architecture, but is not a message queue. Message queues by nature store messages while MQTT does not. In MQTT, if no one is subscribing (or listening) to a topic, it is simply ignored and lost. Messages queues also maintain a client-server topology where one consumer is paired with one producer.

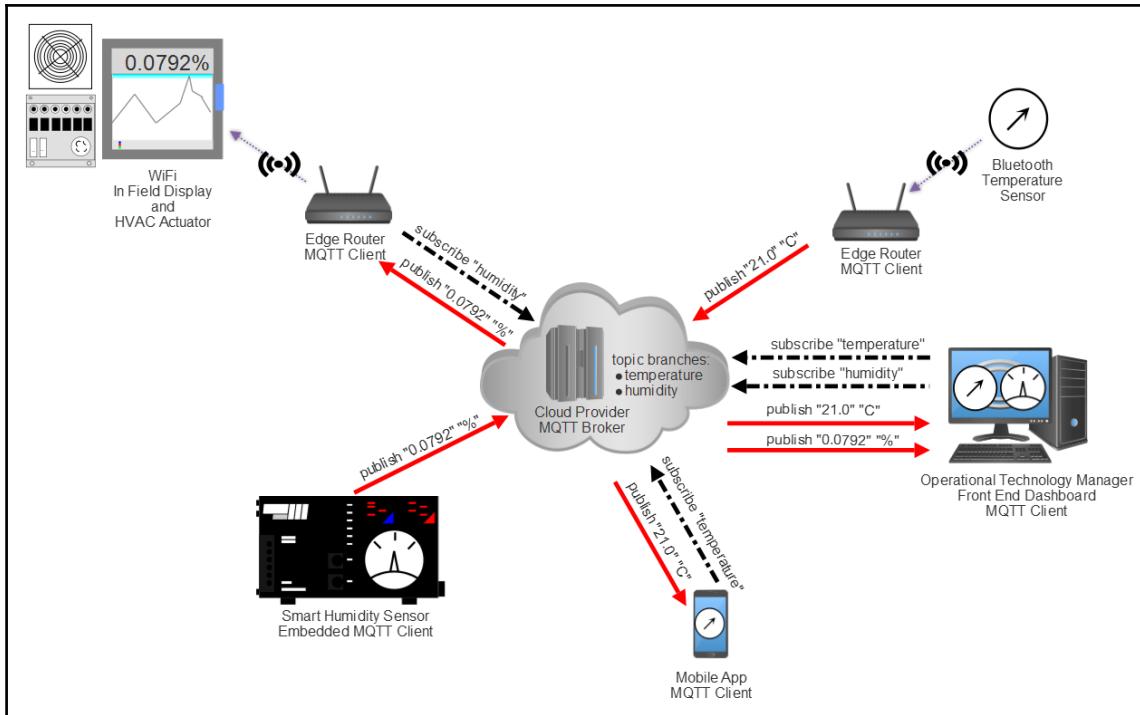


There are *retained* messages in MQTT that will be covered later. A retained message is a single instance of one message that is retained.

A client transmitting a message is called a **publisher**; a client receiving a message is called a **subscriber**. In the center is an MQTT broker who has the responsibility of connecting clients and filtering data. Such filters provide:

- **Subject filtering:** By design, clients subscribe to topics and certain topic branches and do not receive more data than they want. Each published message must contain a topic and the broker is responsible for either re-broadcasting that message to subscribed clients or ignoring it.
- **Content filtering:** Brokers have the ability to inspect and filter published data. Thus, any data not encrypted can be managed by the broker before being stored or published to other clients.
- **Type filtering:** A client listening to a subscribed data stream can apply their own filters as well. Incoming data can be parsed and the data stream either processed further or ignored.

MQTT may have many producers and many consumers, as shown in the following figure:



MQTT publish-subscribe model and topology. Clients run on the edge and publish and/or subscribe to topics that are managed by the MQTT Broker. Here, two topics are considered: humidity and temperature. A client may subscribe to multiple topics. The figure represents smart sensors that include enough resources to manage their own MQTT client and edge routers that provide the MQTT client services on behalf of sensors or appliances that are not MQTT-enabled.



One caveat of the publisher/subscriber model of computing is that both the publisher and subscriber must know the topic branch and data format before initiating transmission.

MQTT successfully decouples publishers from consumers. Since the broker is the governing body between publishers and consumers there is no need to directly identify a publisher and consumer based on physical aspects (such as an IP address). This is very useful in IoT deployments as the physical identity may be unknown or ubiquitous. MQTT and other pub/sub models are also time-invariant. This implies that a message published by one client may be read and responded to at any time by a subscriber. A subscriber could be in a very low power/bandwidth-limited state (for example, Sigfox communication) and respond minutes or hours later to a message. Because of the lack of physical and time relationships, pub/sub models scale very well to extreme capacity.

Cloud-managed MQTT brokers typically can ingest millions of messages per hour and support tens of thousands of publishers.

MQTT is data format agnostic. Any type of data may reside in the payload which is why both the publishers and subscribers must both understand and agree to the data format. It is possible to transmit text messages, image data, audio data, encrypted data, binary data, JSON objects, or virtually any other structure in the payload. JSON text and binary data, however, are the most common data payload types.

The maximum allowable packet size in MQTT is 256 MB, which allows for an extremely large payload.



Note, however, that the maximum data payload size is cloud and broker-dependent. For example, IBM Watson allows for payload sizes up to 128 KB, while Google supports 256 KB. Alternatively, a published message can include a zero-length payload. The payload field is optional. It is advisable to check with your cloud provider to ensure the payload sizes are matched. Failure to do so will result in errors and disconnections from the cloud broker.

MQTT architecture details

MQTT by name is a misnomer. There are no message queues inherent in the protocol. While it is possible to queue messages, it's not necessary and often not done. MQTT is TCP based, and therefore includes some guarantee that a packet is transferred reliably.

MQTT is also an *asymmetric protocol*, whereas HTTP is a *nonsymmetric protocol*. Say node A needs to communicate with node B. An asymmetric protocol between A and B would only require one side (A) to use the protocol, however, all information needed for the reassembly of packets must be contained in the fragmentation header sent by A. Asymmetric systems have one master and one slave (FTP being a classic example). In a symmetric protocol, both A and B would have the protocol installed. A or B can assume the role of master or slave (telnet being a primary example). MQTT has distinct roles which makes sense in sensor/cloud topology.

MQTT can retain a message on a broker indefinitely. This mode of operation is controlled by a flag on a normal message transmission. A retained message on a broker is sent to any client that subscribes to that MQTT topic branch. The message is transmitted immediately to that new client. This allows a new client to receive a status or signal from a newly subscribed topic without waiting. Typically, a client subscribing to a topic may need to wait hours, or even days, before a client publishes new data.

MQTT defines an optional facility called **Last Will and Testament (LWT)**. An LWT is a message a client specifies during the connect phase. The LWT contains the Last Will topic, QoS, and the actual message. If a client disconnects from a broker connection ungracefully (for example, keep-alive timeout, I/O error, or the client closing the session without a disconnect), the broker is then obligated to broadcast the LWT message to all other subscribed clients to that topic.

Even though MQTT is based on TCP, connections can still be lost, especially in a wireless sensor environment. A device may lose power, lose signal strength, or simply crash in the field, and a session will enter a half-open state. Here, the server will believe the connection is still reliable and expect data. To remedy this half-open state, MQTT uses a keep-alive system. Using this system, both the MQTT broker and client have assurance that the connection is still valid even if there hasn't been a transmission for some time. The client sends a PINGREQ packet to the broker, which in turn acknowledges the message with a PINGRESP. A timer is preset on the client and broker side. If a message has not been transmitted from either within a predetermined time limit, a keep-alive packet should be sent. Both a PINGREQ or a message will reset the keep-alive timer.

In the event that a keep-alive is not received and the timer expires, the broker will close the connection and send out the LWT packet to all clients. The client may at some point later attempt to reconnect. In that case, a half-open connection will be closed by the broker and open up a new connection to the client.



The maximum keep alive time is 18 hours, 12 minutes and 15 seconds. Setting the keep-alive internal to 0 will disable the keep-alive functionality. The timer is controlled by the client and can be dynamically changed to reflect sleep modes or changes in signal strength.

While keep-alive assists with broken connections, re-establishing all of a client's subscriptions and QoS parameters can result in unnecessary overhead on a data-capped connection. To alleviate this extra data, MQTT allows for persistent connections. A persistent connection will save the following on the broker side:

- All client's subscriptions
- All QoS messages that were not confirmed by the client
- All new QoS messages missed by the client

This information is referenced by the `client_id` parameter to identify unique clients. A client can request a persistent connection, however, the broker can deny the request and force a clean session to restart. Upon connection, the `cleanSession` flag is used by the broker to allow or deny persistent connections. The client can determine if a persistent connection was stored using the CONNACK message.



Persistent sessions should be used for clients that must receive all messages even when offline. They should not be used in situations where a client only publishes (writes) data to topics.

There are three levels of quality of service in MQTT:

- **QoS-0 (non-assured transmission):** This is the minimal QoS level. This is analogous to a fire-and-forget model detailed in some of the wireless protocols. It is a best-effort delivery process without the receiver acknowledging a message or the sender reattempting transmission.
- **Qos-1 (assured transmission):** This mode will guarantee delivery of the message at least once to the receiver. It may be delivered more than once, and the receiver will send an acknowledgment back with a PUBACK response.
- **QoS-2 (assured service on applications):** This is the highest level of QoS that ensures and informs both the sender and receiver that a message has been transmitted correctly. This mode generates more traffic with a multi-step handshake between the sender and receiver. If a receiver gets a message set to QoS-2, it will respond with a PUBREC message to the sender. This acknowledges the message and the sender will respond with a PUBREL message. PUBREL allows the receiver to safely discard any re-transmissions of the message. The PUBREL is then acknowledged by the receiver with a PUBCOMP. Until the PUBCOMP message is sent, the receiver will cache the original message for safety.

QoS in MQTT is defined and controlled by the sender and each sender can have a different policy.

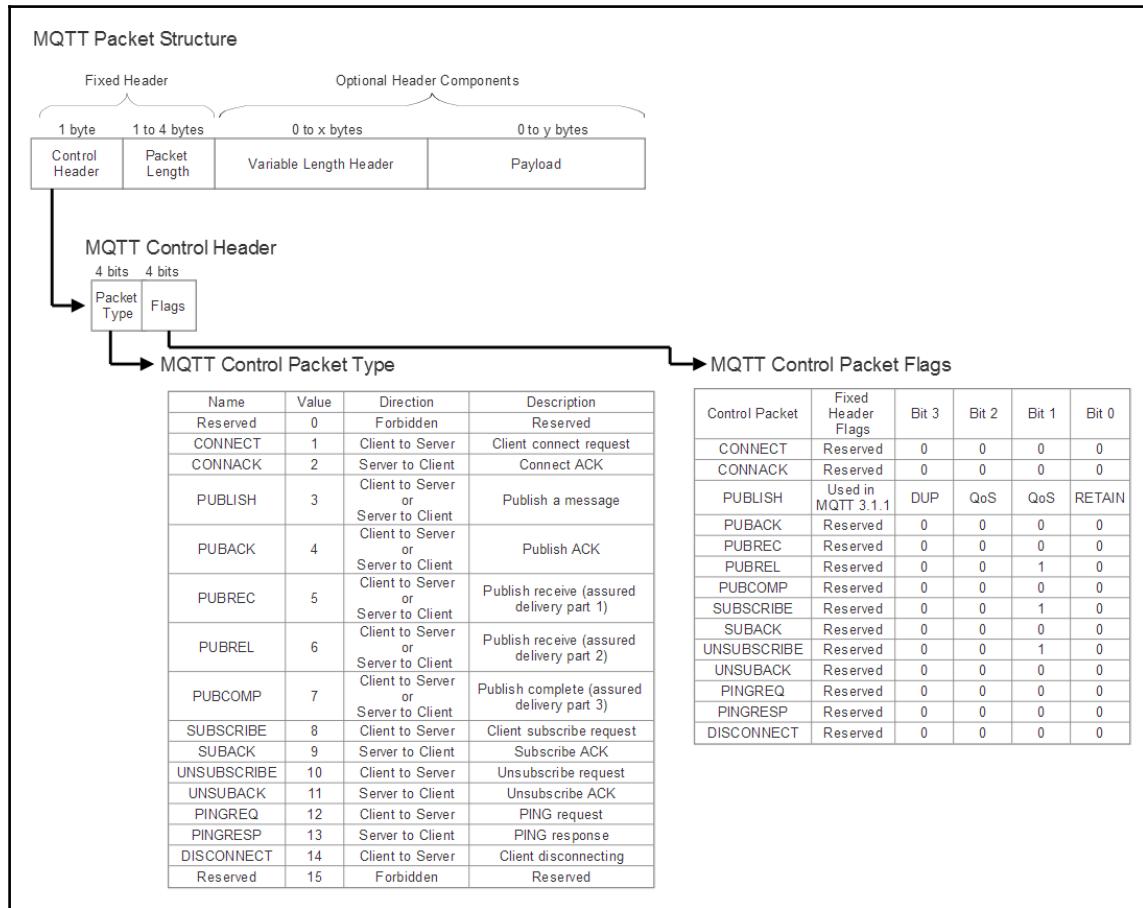
Typical use cases:



- **QoS-0:** Should be used when message queuing isn't needed. It is best for wired connections or when the system is severely constrained on bandwidth.
- **Qos-1:** This should be the default usage. QoS1 is much faster than QoS2 and greatly reduces transmission cost.
- **QoS-2:** For mission-critical applications. Also, cases where re-transmission of duplicate message could cause faults.

MQTT packet structure

An MQTT packet sits on top of the TCP layer of the OSI model network stack. The packet consists of a 2 byte fixed header, which must always be present, a variably sized header (optional) and concludes with the payload (again, optional):



Common packet structure of MQTT

MQTT communication formats

A communication link using MQTT starts with the client sending a CONNECT message to a broker. Only a client can initiate a session and no client may communicate directly with another client. The broker will always respond back to a CONNECT message with a CONNACK response and status code. Once established the connection remains open. The following are the MQTT messages and formats:

- **CONNECT format (client to server):** A typical CONNECT message will contain the following (only the `clientID` is required to initiate a session):

Field	Requirement	Description
<code>clientID</code>	Required	Identifies client to the server. Each client has a unique client ID. It can be between 1 and 23 UTF-8 bytes long.
<code>cleanSession</code>	Optional	0: Server must resume communications with the client. Client and server must save session state after disconnect. 1: Client and server must discard and the previous session and start a new one.
<code>username</code>	Optional	Name used by the server for authentication.
<code>password</code>	Optional	0 to 65536 byte binary password prefixed by 2 bytes of length.
<code>lastWillTopic</code>	Optional	Topic branch to publish will message.
<code>lastWillQos</code>	Optional	2 bits specifying QoS level when publishing will message.
<code>lastWillMessage</code>	Optional	Defines the Will message payload.
<code>lastWillRetain</code>	Optional	Specifies if the Will is to be retained when published.
<code>keepAlive</code>	Optional	Time interval in seconds. The client is responsible for sending a message or a PINGREQ packet before the <code>keepAlive</code> timer expires. The server will disconnect from the network at 1.5x the Keep Alive period. A value of zero (0) will disable the <code>keepAlive</code> mechanism.

- **CONNECT return codes (server to client):** The broker will respond to a CONNECT message with a response code. A designer should be aware that not all connections may be approved by a broker. Response codes are as follows:

Return code	Description
0	Successful connection
1	Connection refused—unacceptable MQTT protocol version
2	Connection refused—client identified is the correct UTF-8, but not allowed by the server
3	Connection refused—server is unavailable
4	Connection refused—bad username or password
5	Connection refused—the client is not authorized to connect

- **PUBLISH format (client to server):** At this point, a client may publish data to a topic branch. Each message contains a topic:

Field	Requirement	Description
packetID	Required	Uniquely identifies the packet in the variable header. Client library responsibility. Always set to zero(0) for QoS-0.
topicName	Required	Topics branch to publish to (such as US/Wisconsin/Milwaukee/temperature).
qos	Required	QoS level 0,1, or 2.
retainFlag	Required	Name used by the server for authentication.
payload	Optional	Data format agnostic payload.
dupFlag	Required	Message is a duplicate and is resent.

- **SUBSCRIBE format (client to server):** The payload of a subscribe packet includes at least one pair of UTF-8 encoded topicIDs and QoS levels. There may be multiple topicIDs subscribed to in this payload to spare the client from multiple broadcasts:

Field	Requirement	Description
packetID	Required	Uniquely identifies the packet in the variable header. Client library responsibility.
topic_1	Required	Subscribed topic branch.
qos_1	Required	QoS level of messages published to topic_1.
topic_2	Optional	Name used by the server for authentication.
qos2	Optional	QoS level of messages published to topic_2.

Wildcards can be used to subscribe to multiple topics in a single message. For these examples, the topic will be the full path of

"{country}/{states}/{cities}/{temperature,humidity}".

- + **Single level wildcard:** Substitutes a single level in a topic string name. For example, US/+Milwaukee will substitute the state level and replace it with all 50 states, Alaska to Wyoming.
- * **Multi-level wildcard:** Replaces multiple levels rather than a single level. It is always the last character in the topic name. For example, US/Wisconsin/# will subscribe to all Wisconsin cities: Milwaukee, Madison, Glendale, Whitefish Bay, Brookfield, and so on.
- \$ **Special topics:** This is a special statistical mode for MQTT brokers. Clients cannot publish to \$ topics. There is no official standard for use at the moment. One model uses \$SYS in the following manner: \$SYS/broker/clients/connected.



An MQTT server should (but this is not explicitly required by specification) support wildcards in the topic name. If it doesn't support wildcards, the server must reject them. Setting the packet ID is the responsibility of the MQTT client library.

Several other messages are in the MQTT specification. More details on the MQTT programming API can be found in OASIS MQTT Version 3.1.1 Standard, <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf>, 2014.

MQTT working example

For a working example, **Google Cloud Platform (GCP)** will be used as the MQTT receiver and ingestor into their cloud. Most MQTT cloud services follow a similar model, so this framework can be used as a reference. We will use open source tools to start an MQTT client and a simple Python example to publish a hello world string to a topics branch. Preliminary steps are needed to start using GCP. Google accounts and payment systems need to be secured before proceeding. Please refer to these instructions regarding getting started with Google IoT Core <https://cloud.google.com/iot/docs/how-tos/getting-started>.

Proceed within GCP to create a device, enable the Google API, create a topics branch, and add a member to the pub/sub publisher.

Google is unique in that it requires strong encryption (TLS) on top of MQTT to encrypt all data packets using **JSON Web Tokens (JWT)** and a certificate agent. Each device will create a public/private key pair. Google ensures each device has a unique ID and key; in the event one is compromised, it will only affect a single node and contain the surface area of an attack.

The MQTT broker starts by importing several libraries. The `paho.mqtt.client` Python library is an Eclipse-sponsored project and is the referenced home of the original IBM MQTT project. Paho is also the core deliverable of the Eclipse M2M Industry Working Group. There are other variants of MQTT message brokers, such as the Eclipse Mosquitto project and Rabbit MQ:

```
#Simple MQTT Client publishing example for Google Cloud Platform
import datetime
import os
import time
import paho.mqtt.client as mqtt
import jwt

project_id = 'name_of_your_project'
cloud_region = 'us-central1'
registry_id = 'name_of_your_registry'
device_id = 'name_of_your_device'
algorithm = 'RS256'
mqtt_hostname = 'mqtt.googleapis.com'
mqtt_port = 8883
ca_certs_name = 'roots.pem'
private_key_file = '/Users/joeuser/mqtt/rsa_private.pem'
```

The next step is authentication with Google through the use of a key. Here we use a JWT object to contain the certificate:

```
#Google requires certificate based authentication using JSON Web Tokens
#(JWT) per device.
#This limits surface area of attacks
def create_jwt(project_id, private_key_file, algorithm):
    token = {
        # The time that the token was issued
        'iat': datetime.datetime.utcnow(),
        # The time the token expires.
        'exp': datetime.datetime.utcnow() +
    datetime.timedelta(minutes=60),
        # Audience field = project_id
        'aud': project_id
    }

    # Read the private key file.
    with open(private_key_file, 'r') as f:
        private_key = f.read()
    return jwt.encode(token, private_key, algorithm)
```

We define several callbacks using the MQTT library such as errors, connections, disconnects, and publishing:

```
#Typical MQTT callbacks
def error_str(rc):
    return '{}: {}'.format(rc, mqtt.error_string(rc))

def on_connect(unused_client, unused_userdata, unused_flags, rc):
    print('on_connect', error_str(rc))

def on_disconnect(unused_client, unused_userdata, rc):
    print('on_disconnect', error_str(rc))

def on_publish(unused_client, unused_userdata, unused_mid):
    print('on_publish')
```

The main structure of an MQTT client follows. First, we register the client as Google prescribes. Google IoT requires a project to be identified, a region, a registry ID, and a device ID. We also skip the username and use the password field via the `create_jwt` method. This is also where we enable SSL encryption in MQTT—many MQTT cloud providers require this provision. After connecting to Google's cloud MQTT server, the program's main loop publishes a simple hello world string to a topics branch which is subscribed to. Also of note is the QoS level that is set in the publish message.

If a parameter is required but not explicitly set in the program, the client library is obligated to use default values (for example, the RETAIN and DUP flags used as default values during the PUBLISH message):

```
def main():
    client = mqtt.Client()
    client_id='projects/{}/locations/{}/registries/{}/devices/{}'
              .format(
                  project_id,
                  cloud_region,
                  registry_id,
                  device_id)) #Google requires this format
    explicitly

    client.username_pw_set(
        username='unused', #Google ignores the user name.
        password=create_jwt( #Google needs the JWT for authorization
            project_id, private_key_file, algorithm))

    # Enable SSL/TLS support.
    client.tls_set(ca_certs=ca_certs_name)
    #callback unused in this example:
    client.on_connect = on_connect
    client.on_publish = on_publish
    client.on_disconnect = on_disconnect

    # Connect to the Google pub/sub
    client.connect(mqtt_hostname, mqtt_port)

    # Loop
    client.loop_start()

    # Publish to the events or state topic based on the flag.
    sub_topic = 'events'
    mqtt_topic = '/devices/{}/{}'.format(device_id, sub_topic)

    # Publish num_messages messages to the MQTT bridge once per second.
    for i in range(1,10):
        payload = 'Hello World!: {}'.format(i)
        print('Publishing message\'{}\{}'.format(payload))
        client.publish(mqtt_topic, payload, qos=1)
        time.sleep(1)

if __name__ == '__main__':
    main()
```

MQTT-SN

A derivative of MQTT is called MQTT-SN (sometimes called MQTT-S) for sensor networks. It keeps to the same philosophy of MQTT as a lightweight protocol for edge devices but is architected specifically for the nuances of a wireless personal area network typical in sensor environments. These traits include support for low-bandwidth links, link failure, short message length, and resource-constrained hardware. MQTT-SN is, in fact, so light that it can be run successfully over BLE and Zigbee.

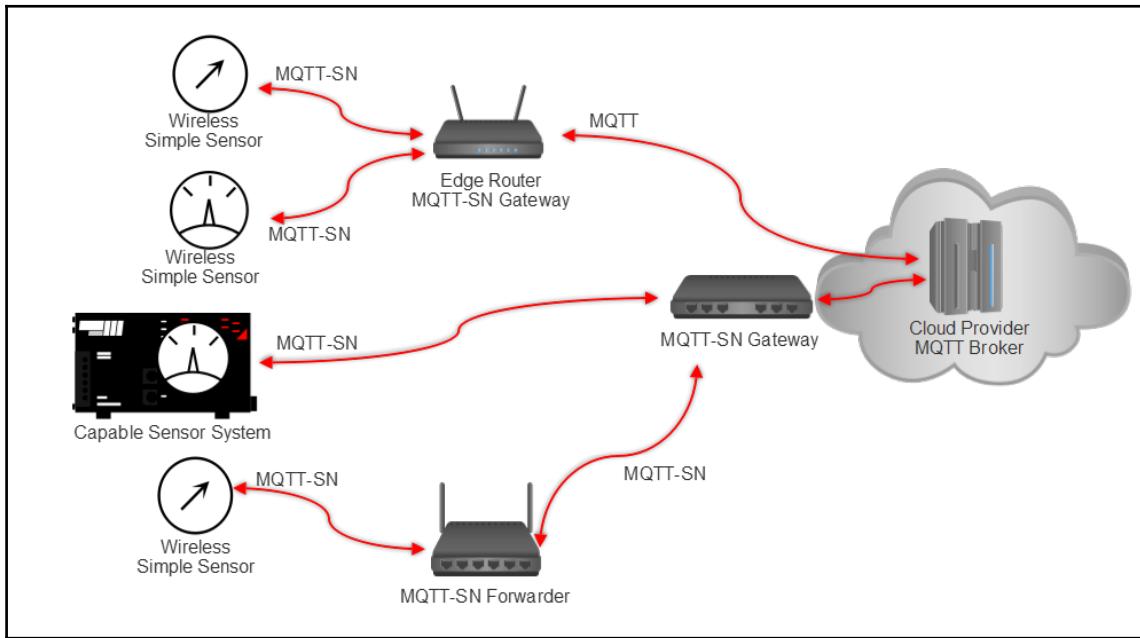
MQTT-SN does not require TCP/IP stack. It can be used over a serial link (preferred way), where a simple link protocol (to distinguish different devices on the line) overhead is really small. Alternatively it can be used over UDP, which is less hungry than TCP.

MQTT-SN architecture and topology

There are four fundamental components in an MQTT-SN topology:

- **Gateways:** In MQTT-SN, a gateway has the responsibility of protocol conversion from MQTT-SN to MQTT and vice versa (although other translations are possible). Gateways can also be aggregating or transparent (covered later in this chapter).
- **Forwarders:** A route between a sensor and an MQTT-SN gateway may take many paths and hop across several routers along the way. Nodes between the source client and the MQTT-SN gateway are called forwarders and simply re-encapsulate MQTT-SN frames into new and unchanged MQTT-SN frames that are sent to the destination until they arrive at the correct MQTT-SN gateway for protocol conversion.
- **Clients:** Clients behave in the same way as in MQTT and are capable of subscribing and publishing data.

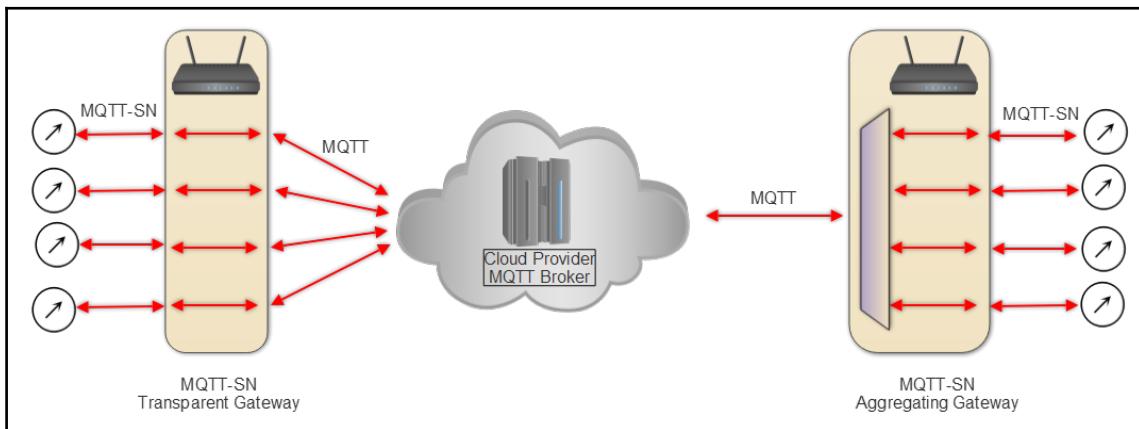
- **Brokers:** Brokers behave in the same way as in MQTT:



MQTT-SN topology. Wireless sensors communicate either to MQTT-SN gateways, which translate MQTT-SN to MQTT, to other protocol forms, or to forwarders that simply encapsulate MQTT-SN frames received into MQTT-SN messages forwarded to a gateway.

Transparent and aggregating gateways

In MQTT-SN, gateways can take on two distinct roles. First, a transparent gateway will manage many independent MQTT-SN streams from sensor devices and convert each stream into an MQTT message. An aggregating gateway will coalesce a number of MQTT-SN streams into a reduced number of MQTT streams sent to the cloud MQTT broker. An aggregating gateway is more complex in design but will reduce the amount of communication overhead and number of simultaneous connections left open on the server. For an aggregating gateway topology to function, the clients need to publish or subscribe to the same topic:



MQTT-SN gateway configurations. A transparent gateway simply performs a protocol conversion on each incoming MQTT-SN stream and has a one-to-one relationship with MQTT-SN connections and MQTT connections to the broker. An aggregating gateway, however, will coalesce multiple MQTT-SN streams into a single MQTT connection to the server.

Gateway advertisement and discovery

Since the topology of MQTT-SN is slightly more complicated than MQTT, a discovery process is used to establish routes through multiple gateways and forwarder nodes. Gateways joining an MQTT-SN topology start by binding to an MQTT broker first. Afterwards, it may issue an ADVERTISE packet to connected clients or other gateways. Multiple gateways can exist on the network, but a client may only connect to a single gateway. The client has the responsibility of storing a list of active gateways and their network addresses. This list is constructed from the various ADVERTISEMENT and GWINFO messages being broadcast.

Because of the new types of gateways and topologies in MQTT-SN, several new messages are available to assist in discovery and advertising:

- **ADVERTISE:** Broadcast periodically from a gateway to advertise its presence.
- **SEARCHGW:** Broadcast by a client when searching for a particular gateway. Part of the message is a `radius` parameter, which states how many hops the SEARCHGW message should follow in a network topology. For example, a value of 1 indicates a single hop in a very dense network where every client is reachable with a single hop.
- **GWINFO:** This is the response by gateways upon receiving a SEARCHGW message. It contains the gateway ID and gateway address, which is only broadcast when the SEARCHGW is sent from a client.

Differences between MQTT and MQTT-SN

The main differences between MQTT-SN and MQTT are as follows:

- There are three CONNECT messages in MQTT-SN versus one in MQTT. The additional two are used to transport the Will topic and Will message explicitly.
- MQTT-SN can run over simplified medium and UDP.
- Topic names are replaced by short, two-byte long topic ID messages. This is to assist with bandwidth constraints in wireless networks.
- Pre-defined topic IDs and short topic names can be used without any registration. To use this feature, both the client and server need to use the same topic ID. Short topic names are short enough to be embedded in the PUBLISH message.
- A discovery procedure is introduced to assist clients and to allow them to find the network addresses of servers and gateways. Multiple gateways may exist in the topology and can be used to load share communication with clients.
- The `cleanSession` is extended to the Will feature. A client subscription can be retained and preserved, but now the Will data is also preserved.
- A revised `keepAlive` procedure is used in MQTT-SN. This is to support sleeping clients in which all messages destined for them are buffered by a server or edge-router and delivered upon awakening.

Constrained Application Protocol

The **Constrained Application Protocol (CoAP)** is the product of the IETF (RFC7228). The IETF **Constrained RESTful Environments (CoRE)** working group created the first draft of the protocol in June 2014 but had worked for several years on its creation. It is specifically intended as a communication protocol for constrained devices. The core protocol is now based on RFC7252. The protocol is unique as it is first tailored for **machine-to-machine (M2M)** communication between edge nodes. It also supports mapping to HTTP through the use of proxies. This HTTP mapping is the on-board facility to get data across the internet.

CoAP is excellent at providing a similar and easy structure of resource addressing familiar to anyone with experience of using the web but with reduced resources and bandwidth demands. A study performed by Colitti et. al demonstrated the efficiency of CoAP over standard HTTP (*Colitti, Walter & Steenhaut, Kris & De, Niccolò. (2017). Integrating Wireless Sensor Networks with the Web.*) CoAP provides a similar functionality with significantly less overhead and power requirements.

Additionally, some implementations of CoAP perform up to 64x better than HTTP equivalents on similar hardware.

	Bytes per transaction	Power	Battery lifetime
CoAP	154	0.744 mW	151 days
HTTP	1451	1.333 mW	84 days

CoAP architecture details

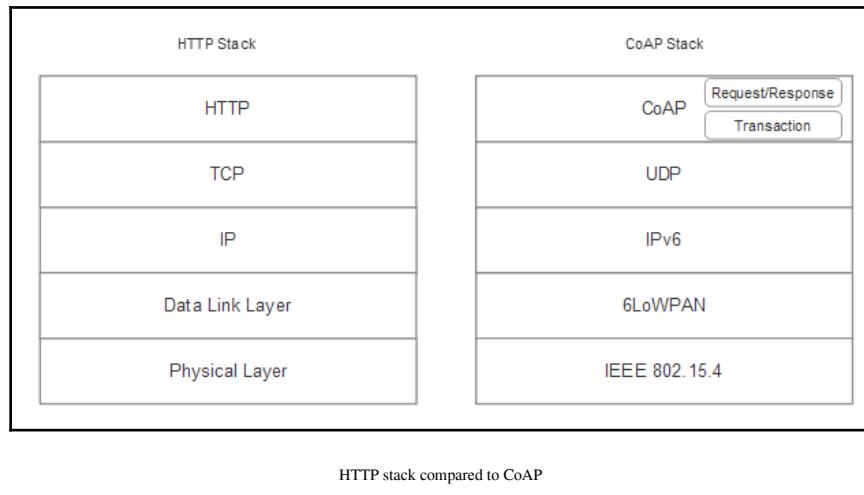
CoAP is based on the concept of mimicking and replacing heavy HTTP abilities and usage with a lightweight equivalent for the IoT. It is not an HTTP replacement as it does lack features; HTTP requires more powerful and service-orientated systems. CoAP features can be summarized as follows:

- HTTP-like
- Connection-less protocols
- Security through DTLS rather than TLS in a normal HTTP transmission
- Asynchronous message exchanges
- Lightweight design and resource requirements and low header overhead
- Support for URI and content-types
- Built upon UDP versus TCP/UDP for a normal HTTP session
- A stateless HTTP mapping allowing for proxies to bridge to HTTP sessions

CoAP has two basic layers:

- **Request/Response layer:** Responsible for sending and receiving RESTful-based queries. REST queries are piggybacked on CON or NON messages. A REST response is piggybacked on the corresponding ACK message.

- **Transactional layer:** Handles single message exchanges between endpoints using one of the four basic message types. The transaction layer also supports multicasting and congestion control:



CoAP shares context, syntax, and usage similarly to HTTP. Addressing in CoAP is also styled like HTTP. An address extends to the URI structure. As in HTTP URIs, the user must know the address beforehand to gain access to a resource. At the top-most level, CoAP uses requests such as GET, PUT, POST, and DELETE, as in HTTP. Similarly, response codes mimic HTTP, such as:

- 2.01: Created
- 2.02: Deleted
- 2.04: Changed
- 2.05: Content
- 4.04: Not found (resource)
- 4.05: Method not allowed

The form of a typical URI in CoAP would be:

```
coap://host[:port]/[path][?query]
```

A CoAP system has seven main actors:

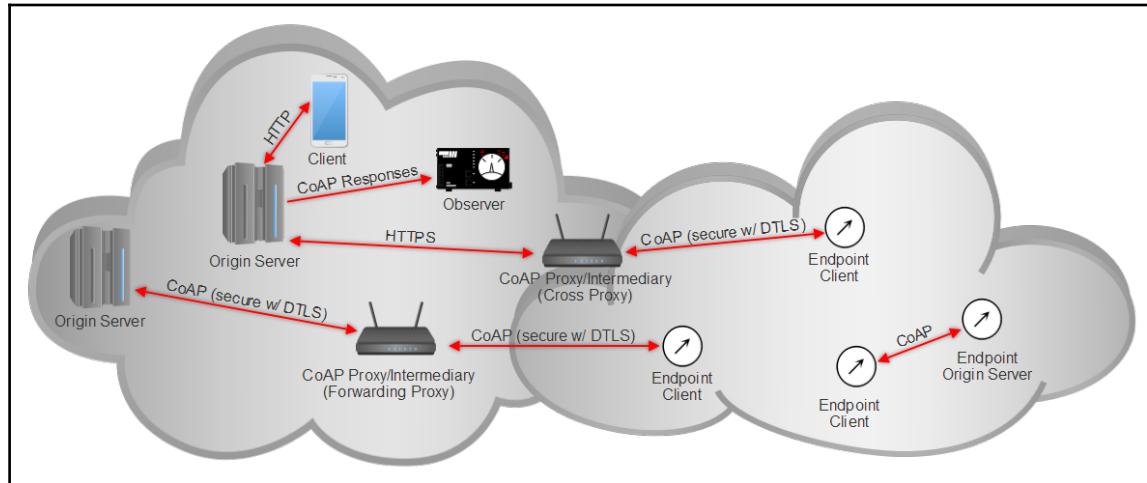
- **Endpoints:** These are the sources and destinations of a CoAP message. The specific definition of an endpoint depends on the transport being used.
- **Proxies:** A CoAP endpoint that is tasked by CoAP clients to perform requests on its behalf. Reducing network load, access sleeping nodes, and providing a layer of security are some of the roles of a proxy. Proxies can be explicitly selected by a client (forward-proxying) or can be used as in-situ servers (reverse-proxying). Alternatively, a proxy can map from one CoAP request to another CoAP request or even translate to a different protocol (cross-proxying). A common situation is an edge router proxying from a CoAP network to HTTP services for cloud-based internet connections.
- **Client:** The originator of a request. The destination endpoint of a response.
- **Server:** The destination endpoint of a request. The originator of a response.
- **Intermediary:** A client acting as both a server and client towards an origin server. A proxy is an intermediary.
- **Origin servers:** The server on which a given resource resides.
- **Observers:** An observer client can register itself using a modified GET message. The observer is then connected to a resource and if the state of that resource changes, the server will send a notification back to the observer.



Observers are unique in CoAP and allow a device to watch for changes to a particular resource. In essence, this is similar to the MQTT subscribe model where a node will subscribe to an event.

Below is an example of CoAP architecture. Being a lightweight HTTP system allows for CoAP clients to communicate to one another or services in the cloud supporting CoAP. Alternatively, a proxy can be used to bridge to an HTTP service in the cloud. CoAP endpoints can establish relationships with each other, even at the sensor level. Observers allow for subscription-like attributes to resource that change in a similar manner to MQTT. The graphic also illustrates origin servers holding the resource being shared.

The two proxies allow for CoAP to perform HTTP translation or allow forwarding requests on behalf of a client.



CoAP Architecture.



CoAP makes use of port 5683. This port must be supported by a server that offers resources since the port is used for resource discovery. Port 5684 is used when DTLS is enabled.

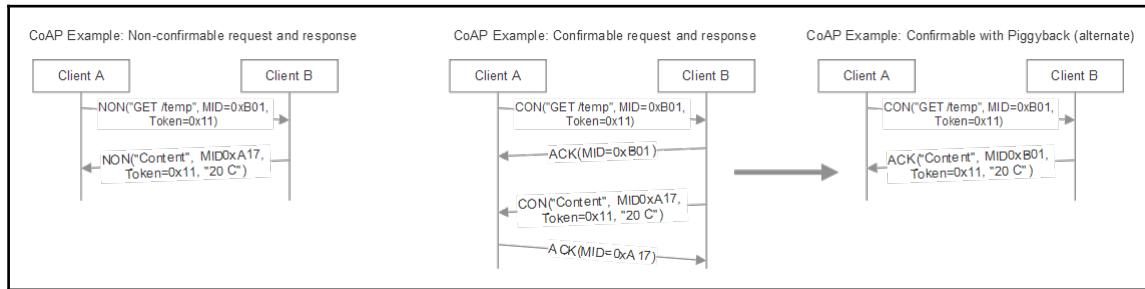
CoAP Messaging Formats

Protocols based on UDP transport imply that the connection may not be inherently reliable. To compensate for reliability issues, CoAP introduces two message types that differ by either requiring acknowledgment or not. An additional feature of this approach is that messages can be asynchronous.

In total there are only four messages in CoAP:

- **Confirmable (CON):** Requires an ACK. If a CON message is sent an ACK must be received within a random time interval between ACK_TIMEOUT and (ACK_TIMEOUT * ACK_RANDOM_FACTOR). If the ACK is not received, the sender transmits the CON message over and over at exponentially increasing intervals until it receives the ACK or a RST. This essentially is the CoAP form of congestion control. There is a maximum number of attempts set by MAX_RETRANSMIT. This is the resiliency mechanism to compensate for the lack of resiliency in UDP.
- **Non-confirmable (NON):** Requires no ACK. Essentially a fire-and-forget message or broadcast.
- **Acknowledgement (ACK):** Acknowledges a CON message. The ACK message can piggyback along with other data.
- **Reset (RST):** Indicates that a CON message has been received but the context is missing. The RST message can piggyback along with other data.

CoAP is a RESTful design using request/response messages piggybacked on CoAP messages. This allows for greater efficiency and bandwidth preservation, as shown in the following figure:



CoAP NON and CON messaging.

The graphic shows three examples of CoAP non-confirmable and confirmable request/response transactions. These are enumerated and described as follows:

- **Non-confirmable request/response (left):** A message broadcast between client A and B using the typical HTTP GET construct. B reciprocates with Content data sometime later and returns the temperature of 20 degrees celsius.

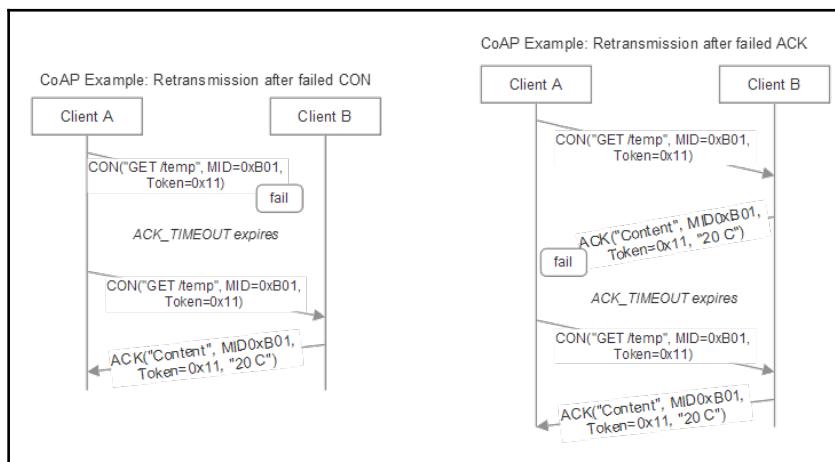
- Confirmable request/response (middle):** Included is the Message-ID, a unique identifier for each message. The Token represents a value that must match during the duration of the exchange.
- Confirmable request/response (right):** Here the message is confirmable. Both client A and B will wait for an ACK after each message exchange. To optimize communication, Client B can elect to piggyback the ACK with the returned data as shown in the far right.

The actual log of CoAP transactions can be seen in the Copper Firefox extension on Firefox version 55:

Time	CoAP Message	MID	Token	Options	Payload
9:09:50 PM	CON-GET	12514	(0)	empty	Uri-Path: .well-known/core, Block2: 0/0/64
9:09:50 PM	ACK-2.05 Content	12514		Content-Format: 40, Block2: 0/1/64, Size2: 1918	</obs>;obs;rt="observe";title="Observable resource which changes
9:09:50 PM	CON-GET	12515	(0)	empty	Uri-Path: .well-known/core, Block2: 1/0/64
9:09:50 PM	ACK-2.05 Content	12515		Content-Format: 40, Block2: 1/1/64	every 5 seconds", </obs-pumping>;obs;rt="observe";title="Observ
9:09:50 PM	CON-GET	12516	(0)	empty	Uri-Path: .well-known/core, Block2: 2/0/64
9:09:50 PM	ACK-2.05 Content	12516		Content-Format: 40, Block2: 2/1/64	able resource which changes every 5 seconds", </separate>;title="R
9:09:50 PM	CON-GET	12517	(0)	empty	Uri-Path: .well-known/core, Block2: 3/0/64
9:09:50 PM	ACK-2.05 Content	12517		Content-Format: 40, Block2: 3/1/64	esource which cannot be served immediately and which cannot be a
9:09:50 PM	CON-GET	12518	(0)	empty	Uri-Path: .well-known/core, Block2: 4/0/64
9:09:50 PM	ACK-2.05 Content	12518		Content-Format: 40, Block2: 4/1/64	cknowledged in a piggy-backed way", </large-create>;rt="block";ti

Copper CoAP log. Here we see several CON-GET client initiated messages to californium.eclipse.org:5683. The URI path points to coap://californium.eclipse.org:5683/.well-known/core. The MID increments with each message while the token is unused and optional.

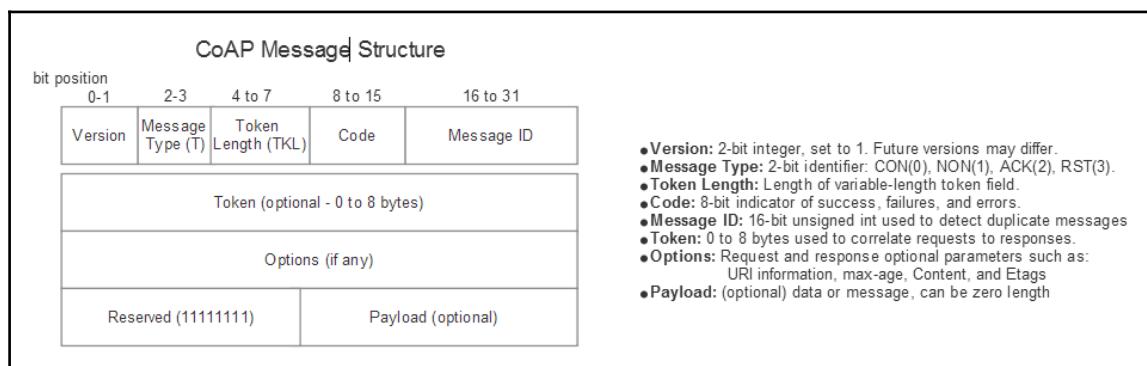
The retransmission process is illustrated in the following figure:



CoAP retransmission mechanism. To account for the lack of resiliency in UDP, CoAP uses a timeout mechanism when communicating with confirmable messages. If the timeout expires, either sending a CON message or receiving the ACK, the sender will retransmit the message. The sender is responsible for managing the timeout and retransmitting up to a maximum number of retransmissions. Note the retransmission of the failed ACK reuses the same Message_ID.

While other messaging architecture requires a central server to propagate messages between clients, CoAP allows messages to be sent between any CoAP client including sensors and servers. CoAP includes a simple caching model. Caching is controlled through response codes in the message header. An option number mask will determine if it is a cache key. The `Max_Age` option is used to control cache element lifetimes and ensure freshness of the data. That is, `Max_Age` sets the maximum time a response can be cached for before it must be refreshed. `Max_Age` defaults to 60 seconds and can span up to 136.1 years. Proxies play a role in caching; for example, a sleeping edge sensor may use a proxy to cache data and to preserve power.

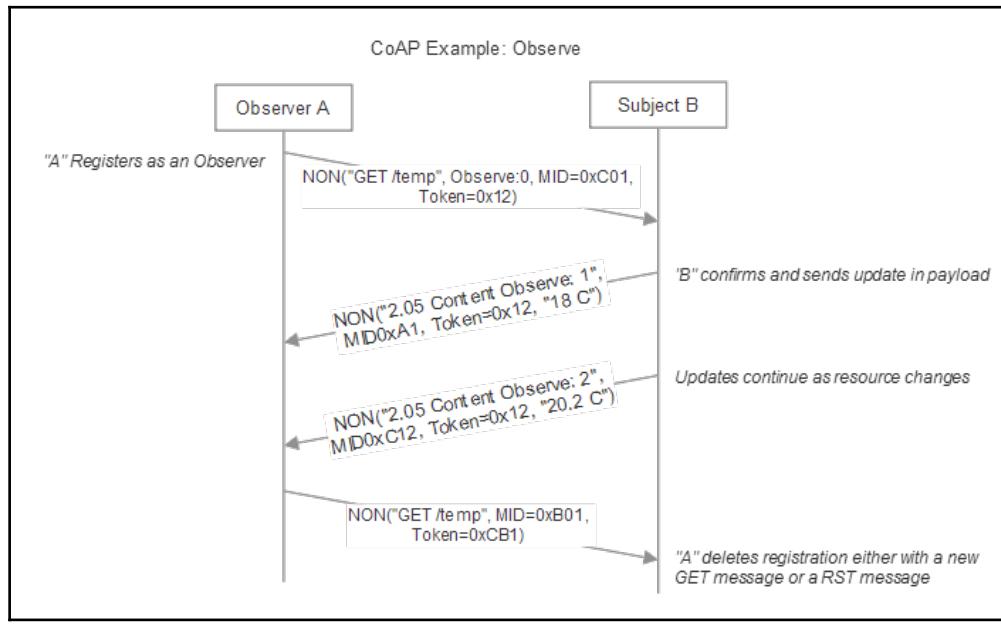
The CoAP message header is uniquely designed for maximum efficiency and bandwidth preservation. The header is four bytes in length with a typical request message taking only 10 to 20-byte headers. This is typically 10x smaller than an HTTP header. The structure consists of Message Type Identifiers (T), which must be included in each header along with an associated unique Message-ID. The **Code** field is used to signal errors or success states across channels. After the header, all other fields are optional and include variable length Tokens, Options, and Payload:



CoAP message structure

UDP also can cause duplicate messages to arrive for both CON and NON transmissions. If identical `Message_IDs` are delivered to a recipient within the prescribed `EXCHANGE_LIFETIME`, a duplicate is said to exist. This clearly can occur, as shown in the previous figures, when an ACK is missing or dropped and the client retransmits the message with the same `Message_ID`. The CoAP specification states the recipient should ACK each duplicate message it receives but should only process one request or response. This rule may be relaxed if a CON message transports a request that is idempotent.

As mentioned, CoAP allows for the role of an observer in a system. This is unique as it allows CoAP to behave in a similar manner to MQTT. The observation process allows a client to register for observation and the server will notify the client whenever the resource being monitored changes state. The duration of observation can be defined during registration. Additionally, the observation relationship ends when the initiating client sends a RST or another GET message:



CoAP observer registration and update process

As mentioned previously, there is no inherent authentication or encryption in the CoAP standard, rather, the user must rely on DTLS to provide that level of security. If DTLS is used then an example URI would be:

```
//insecure
coap://example.net:1234/~temperature/value.xml

//secure
coaps://example.net:1234/~temperature/value.xml
```

CoAP offers resource discovery mechanisms as well. Simply sending a GET request to `/well-known/core` will disclose a list of known resources on the device. Additionally, query strings can be used in the request for applying specific filters.

CoAP usage example

CoAP is lightweight and its implementation both on a client and server basis should take few resources. Here, we use the Python-based aiocoap library. More can be read on aiocoap in: *Amsüss, Christian, and Wasilak, Maciej. aiocoap: Python CoAP Library. Energy Harvesting Solutions, 2013–.* <http://github.com/chrysn/aiocoap/>. Many other free CoAP clients and servers exist with several written in low-level C code for extremely constrained sensor environments. Here, we use a Python environment for brevity.

The client implementation is:

```
#!/usr/bin/env python3
import asyncio          #necessary for asynchronous processing in Python
from aiocoap import *   #using the aiocoap library
```

The following is main loop for the client. The client uses PUT to broadcast the temperature to known URI:

```
async def main():
    context = await Context.create_client_context()

    await asyncio.sleep(2)                                #wait 2 seconds after
    initialization

    payload = b"20.2 C"
    request = Message(code=PUT, payload=payload)

    request.opt.uri_host = '127.0.0.1'                  #URI for localhost address
    request.opt.uri_path = ("temp", "celcius")           #URI for path to
    /temp/celcius

    response = await context.request(request).response
    print('Result: %s\n%s' %(response.code, response.payload))

if __name__ == "__main__":
    asyncio.get_event_loop().run_until_complete(main())
```

The server implementation is:

```
#!/usr/bin/env python3
import asyncio          #necessary for asynchronous
processing in Python
import aiocoap.resource as resource      #using aiocoap library
import aiocoap
```

The following code illustrates the services for PUT and GET methods:

```
class GetPutResource(resource.Resource):  
  
    def __init__(self):  
        super().__init__()  
        self.set_content(b"Default Data (padded)"  
  
    def set_content(self, content):                      #Apply padding  
        self.content = content  
        while len(self.content) <= 1024:  
            self.content = self.content + b"0123456789\n"  
  
    async def render_get(self, request):                 #GET handler  
        return aiocoap.Message(payload=self.content)  
  
    async def render_put(self, request):                 #PUT handler  
        print('PUT payload: %s' % request.payload)  
        self.set_content(request.payload)                #replaces set_content with  
        received payload  
        return aiocoap.Message(code=aiocoap.CHANGED, payload=self.content) #set  
        response code to 2.04
```

The main loop is:

```
def main():  
    root = resource.Site()                         #root element that contains all resources  
    found on server  
  
    root.add_resource('.well-known', 'core'),          #this is the  
    typical .well-known/core  
    resource.WKCResource(root.get_resources_as_linkheader)) #resource list for  
.well-known/core  
    root.add_resource('temp', 'celcius'), GetPutResource())) #adds the  
    resource /tmp/celcius  
  
    asyncio.Task(aiocoap.Context.create_server_context(root))  
  
    asyncio.get_event_loop().run_forever()  
  
if __name__ == "__main__":  
    main()
```

Other protocols

There are many other messaging protocols in use or proposed for IoT and M2M deployments. By far the most prevalent are MQTT and CoAP; the next sections explore a few alternatives for specific use cases.

STOMP

STOMP stands for Simple (or Streaming) Text Message-Oriented Middleware Protocol. It is a text-based protocol designed by Codehaus to operate with message-oriented middleware. A broker developed in one programming language can receive messages from a client written in another. The protocol has similarities to HTTP and operates over TCP. STOMP consists of a frame header and a frame body. The current specification is STOMP 1.2, dated October 22, 2012, and is available under a free license.

It is different from many protocols presented as it does not deal with subscription topics or queues. It simply uses HTTP-like semantics such as SEND with a destination string. A broker must dissect the message and map to a topic or queue for the client. A consumer of the data will SUBSCRIBE to the destinations provided by the broker.

STOMP has clients written in python (`Stomp.py`), TCL (`tStomp`), and Erlang (`stomp.erl`). Several servers have native STOMP support, such as RabbitMQ (via a plugin), and some servers have been designed in specifics languages (Ruby, Perl, or OCaml).

AMQP

AMQP stands for **Advanced Message Queuing Protocol**. It is a hardened and proven MOM protocol used by massive data sources such as JP Morgan Chase in processing over 1 billion messages per day, and the Ocean Observatory Initiative in collecting over 8 terabytes of oceanographic data each day. It was originally devised at JP Morgan Chase in 2003, who led the creation of a working group of 23 companies in 2006 chartered with the architecture and governance of the protocol. In 2011, the working group was merged into the OASIS group where it is currently hosted.

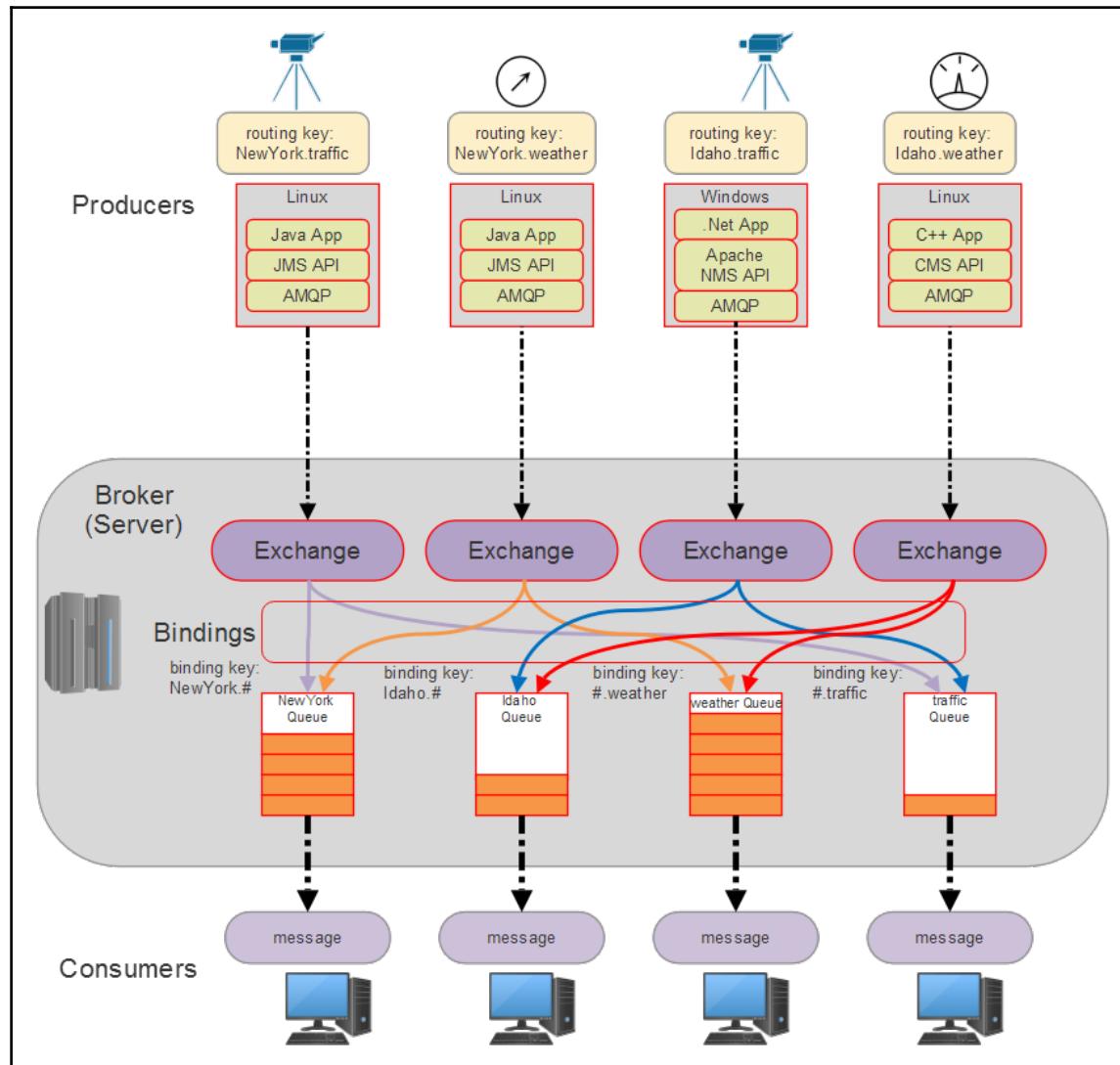
Today it is well established in banking and credit transaction industries but has a place in IoT as well. Furthermore, AMQP is standardized by ISO and IEM as ISO/IEC 19464:2014. A formal AMQP working group can be found at www.amqp.org.

The AMQP protocol resides on top of the TCP stack and uses port 5672 for communication. Data is serialized over AMQP, meaning that messages are broadcast in unit frames. Frames are transmitted in virtual channels identified with a unique `channel_id`. Frames consist of headers, `channel_ids`, payload information, and footers. A channel, however, can only be associated with a single host. Messages are assigned a unique global identifier.

AMQP is a flow-controlled, message-orientated communication system. It is a wire-level protocol and a low-level interface. A wire protocol refers to APIs immediately above the physical layer of a network. A wire-level API allows for different messaging services such as .NET (NMS) and Java (JMS) to communicate with each other. Similarly, AMQP attempts to decouple publishers from subscribers. Unlike MQTT it has mechanisms for load balancing and formal queuing. A well-used protocol that is based on AMQP is RabbitMQ. RabbitMQ is an AMQP message broker written in Erlang. Additionally, several AMQP clients are available, such as the RabbitMQ client written in Java, C#, Javascript, and Erlang, as well as Apache Qpid written in Python, C++, C#, Java, and Ruby.

One or more virtual hosts with their own namespaces, exchanges, and message queues will reside on a central server(s). Producers and consumers subscribe to the exchange service. The exchange service receives messages from a publisher and routes the data to an associated queue. This relationship is called a **binding** and the binding can either be direct to one queue or fanned out to multiple queues (as in a broadcast). Alternatively, the binding can associate one exchange with one queue using a routing key; this is formally called a **direct exchange**. Another type of exchange is the topic exchange.

Here a pattern is used to wildcard a routing key (such as `*.temp.#` matches `idaho.temp.celsius` and `wisconsin.temp.fahrenheit`):



AMQP architectural topology. There are producers and consumers in a typical AMQP implementation. Producers can use different languages and namespaces, as AMQP is language-agnostic given its APIs and a wire protocol. The broker resides in the cloud and provides exchanges for each producer. Exchanges route messages to proper queues based on binding rules. The queues are message buffers that produce messages to awaiting consumers.

The network topology of an AMQP deployment is hub-and-spoke with the ability for hubs to communicate with one another. AMQP consists of nodes and links. A node is a named source or a sink of messages. A message frame moves between nodes across unidirectional links. If a message is passed through a node, the global identifier is unchanged. If the Node performs any transformation, a new ID is assigned. A link has the ability to filter messages. There are three distinct messaging patterns that can be used in AMQP:

- **Asynchronous Directed Messages:** Message is transmitted without requiring a receiver acknowledgment.
- **Request/Reply or Pub/Sub:** Similar to MQTT with a central server acting as a pub/sub service.
- **Store and Forward:** This is used for hub relaying, where a message is sent to an intermediate hub and then sent on towards its destination.

A basic directed exchange written in Python is shown here, using RabbitMQ and the pika Python libraries. Here we create a simple direct exchange called **Idaho** and bind it to a queue called **weather**:

```
#!/usr/bin/env python
#AMQP basic Python example the pika Python library

from pika import BlockingConnection, BasicProperties, ConnectionParameters

#initialize connections
connection = BlockingConnection(ConnectionParameters('localhost'))
channel = connection.channel()
channel.exchange_declare(exchange='Idaho', type='direct')      #declare a
direct exchange
channel.queue_declare(queue='weather')                          #declare the
queue

channel.queue_bind(exchange='Idaho', queue='weather', routing_key='Idaho')
#bindings

#produce the message
channel.basic_publish(exchange='Idaho', routing_key='Idaho', body='new
important task')

#consume the message
method_frame, header_frame, body = ch.basic_get('weather')

#acknowledge
channel.basic_ack(method_frame.delivery_tag)
```

Protocol summary and comparison

A summary and comparison of the various protocols is now given. It should be noted that there are exceptions to some of these categories. For example, while MQTT offers no built-in security provisioning, it can be layered at the application level. In all cases, there are exceptions and the table is built from the formal specifications:

	MQTT	MQTT-SN	CoAP	AMQP	STOMP	HTTP/RESTful
Model	MOM pub/sub	MOM pub/sub	RESTful	MOM	MOM	RESTful
Discovery protocol	No	Yes (via gateways)	Yes	No	No	Yes
Resource demands	Low	Very Low	Very Low	High	Medium	Very High
Header Size (bytes)	2	2	4	8	8	8
Average power usage	Lowest	Low	Medium	High	Medium	High
Authentication	No (SSL/TLS)	No (/TLS)	No (DTLS)	Yes	No	Yes (TLS)
Encryption	No (SSL/TLS)	No (SSL/TLS)	No (DTLS)	Yes	No	Yes (TLS)
Access controls	No	No	No (proxy)	Yes	No	Yes
Communication overhead	Low	Very Low	Very Low	High	High, verbose	High
Protocol complexity	Low	Low	Low	High	Low	Very High
TCP/UDP	TCP	TCP/UDP	UDP	TCP/UDP	TCP	TCP
Broadcasting	Indirect	Indirect	Yes	No	No	No
Quality of Service	Yes	Yes	With CON messages	Yes	No	No

Summary

MQTT, CoAP, and HTTP are by far the predominant IoT internet protocols in the industry with support from nearly every cloud provider. MQTT and MQTT-SN provide a scalable and efficient pub/sub model of data communication while CoAP provides all the relevant features of an HTTP RESTful model without the overhead. An architect must consider the overhead, power, bandwidth, and resources necessary to support a given protocol and have enough forward vision to ensure the solution scales. Now that a transport method has been defined to marshal data to the internet we can examine what to do with that data. The next chapter will concentrate on cloud and fog architecture, from the founding principals to more advanced configurations.

10

Cloud and Fog Topologies

Without the cloud, the IoT growth and market would be non-existent. Essentially, billions of endpoint devices that were historically dumb and not connected would need to manage themselves without the ability to share data or aggregate data. Billions of small embedded systems add no marginal value for customers. The value of the IoT is in the data it produces—not at a single endpoint but in thousands or millions of endpoints. The cloud provides the ability to have simple sensors, cameras, switches, beacons, and actuators participate in a common language with each other. The cloud is the common denominator of the data currency.

The ubiquitous cloud metaphor refers to an infrastructure of computing services that are generally on-demand. The pool of resources (computing, networking, storage, and the associated software services) can dynamically scale up or down based on load average or quality of service. Clouds are typically large data centers that provide outward facing services to customers on a pay-for-use model. These centers provide the illusion of a single cloud resource while in fact there may be many geographically dispersed resources being used. This gives the user a sense of location independence. Resources are elastic (meaning scalable) and services are pay-for-use, generating a recurring revenue stream for the provider. Services that run in the cloud differ in their construction and deployment from traditional software. Cloud-based applications can be developed and deployed faster and with fewer degrees of environmental variability. Thus, cloud deployment enjoys rapid feature velocity.

There are accounts that the first description of the cloud originated at Compaq in the mid-1990s, where technology futurists predicted a computing model that moved computing to the web versus on host platforms. Essentially, this was the basis of cloud computing, but it wasn't until the advent of certain other technologies that cloud computing became practical for the industry. The telecommunications industry traditionally was built on a point-to-point system of circuits. The creation of VPNs allows for secure and controlled access to clusters and has enabled private-public cloud hybrids to exist.

This chapter studies cloud architecture and the following areas:

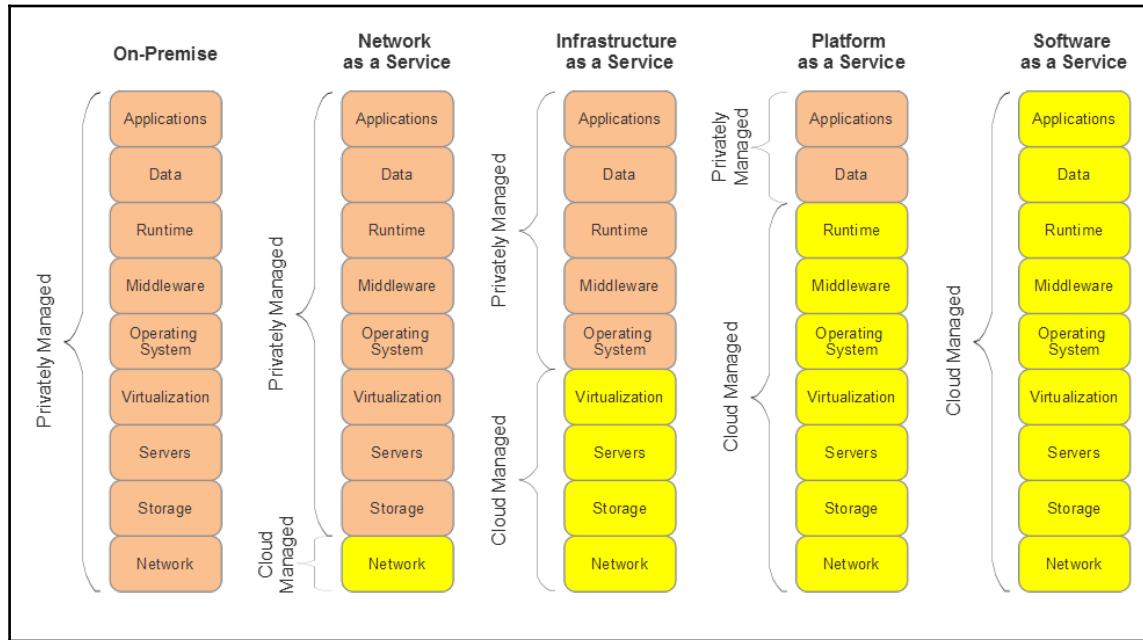
- A formal definition of cloud topologies and vernacular
- An architectural overview of OpenStack cloud
- A study of the fundamental problem with cloud-only architecture
- Fog computing overview
- OpenFog reference architecture
- Fog computing topologies and use cases

Throughout the chapter, several use cases will be talked about so you can understand the impact of big data semantics on IoT sensor environments.

Cloud services model

Cloud providers typically support a range of **Everything as a Service (XaaS)** products. That is, as a pay-for-use software service. Services include **Networking as a Service (NaaS)**, **Software as a Service (SaaS)**, **Platform as a Service (PaaS)**, and **Infrastructure as a Service (IaaS)**. Each model introduces more and more cloud vendor services. These service offerings are the value-add of cloud computing. At a minimum, these services should offset the capital expense a customer faces for purchasing and maintaining such data center equipment and replace it as an operational expense. The standard definition of cloud computing can be found via the National Institute of Standards and Technology: Peter M. Mell and Timothy Grance. 2011. SP 800-145. the NIST Definition of Cloud Computing. Technical Report. NIST, Gaithersburg, MD, United States.

The following figure illustrates the differences in management of cloud models that will be described in the subsequent sections:



Cloud architecture models. On-premises is where all services, infrastructure, and storage are managed by the owner.

NaaS includes services like SDP and SDN. IaaS pushes the hardware systems and storage to the cloud. PaaS includes the infrastructure but also manages the OS and system runtime or containers in the cloud. Finally, SaaS pushes all services, infrastructure, and services to the cloud provider.

NaaS

Services such as **Software-Defined Networking (SDN)** and **Software-Defined Perimeters (SDP)** are typical of NaaS. These products are cloud-managed and organized mechanisms for providing overlay networks and enterprise security. Rather than building a worldwide infrastructure and capital to support a corporation's communications, a cloud approach can be used to form a virtual network. This allows for the network to optimally scale resources up or down based on demand, and new network features can be purchased and deployed rapidly. This topic will be covered in depth in the related SDN chapter:

SaaS

SaaS is the foundation of cloud computing. A provider usually has applications or services on offer that expose themselves to the end user through clients such as mobile devices, thin clients, or frameworks on other clouds. From a user point of view, the SaaS layer is virtually running on their client. This software abstraction has enabled the industry to achieve substantial growth in the cloud. SaaS services include such name brand appliances as Google Apps, Salesforce, and Microsoft Office 365.

PaaS

PaaS refers to the underlying hardware and lower-layer software facilities provided by the cloud. In this case, the end user simply uses a provider's data center hardware, OS, middleware, and assorted frameworks to host their private application or services.

Middleware may be composed of database systems. Many industries have been built using cloud provider commodity hardware such as Swedbank, Trek Bicycles, and Toshiba.

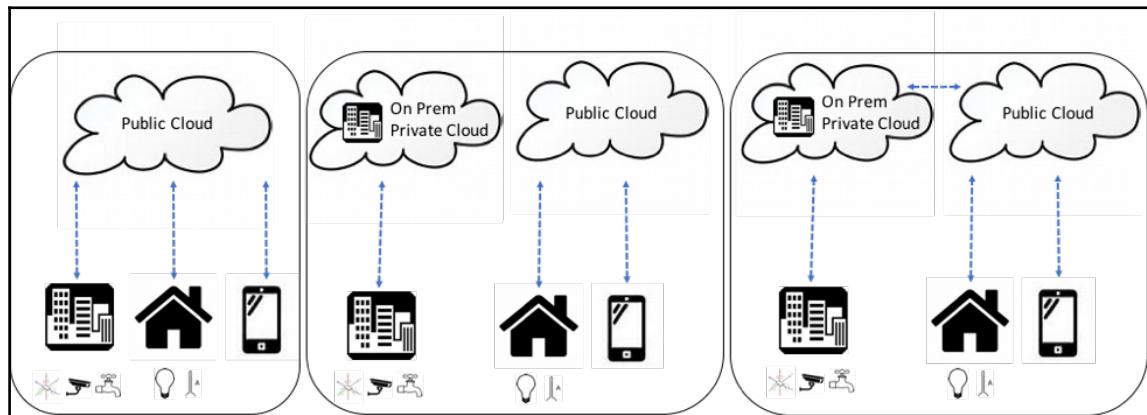
Examples of public PaaS providers are IBM Bluemix, Google App Engine, and Microsoft Azure. The value of a PaaS deployment versus IaaS is the fact that you get the benefits of scalability and OPEX with a cloud infrastructure, but you also have proven middleware and operating systems from the provider. This is the realm of systems such as Docker where software is deployed as containers. If your overall application stays within the constraints of the vendor provided framework, and infrastructure you can achieve a faster time to market since most of the componentry, OS, and middleware are guaranteed to be available.

IaaS

IaaS was the original concept of cloud services. In this model, the provider builds scalable hardware services in the cloud and provides a modicum of software frameworks to build client virtual machines. This offers the most flexibility in deployment but requires more lifting on the part of the customer.

Public, private, and hybrid cloud

Within the cloud environment stand three different models of cloud topologies that are generally used: private cloud, public cloud, and hybrid cloud. Regardless of the model, cloud frameworks should all provide the ability to dynamically scale, develop, and deploy rapidly, and have the appearance of locality regardless of proximity:



Left: public cloud. Middle: private vs. public cloud. Right: hybrid cloud.

Private clouds also imply on-premise managed components. Modern enterprise systems tend to use a hybrid architecture to ensure the safety of mission-critical applications and data on-premise, and use the public cloud for connectivity, deployment ease, and rapid development.

Private cloud

In a private cloud, the infrastructure is provisioned for a single organization or corporation. There is no concept of resource sharing or pooling outside of the owner's own infrastructure. Within the premises, sharing and pooling are common. A private cloud exists for a number of reasons, including security and assurance. That is, to guarantee information is confined solely to systems managed by the customer. To be considered a cloud, however, certain aspects of cloud-like services must exist, such as virtualization and load balancing. A private cloud may be on-premises or it may be dedicated machinery provided by a third party exclusively for their use.

Public cloud

A public cloud is the opposite situation. Here, the infrastructure is provisioned on-demand for a multitude of customers and applications. The infrastructure is a pool of resources anyone can use at any time as part of their service-level agreements. The advantage here is that the sheer scale of cloud data centers allows for unprecedented scalability for many customers, who are only limited to how much of the service they wish to purchase.

Hybrid cloud

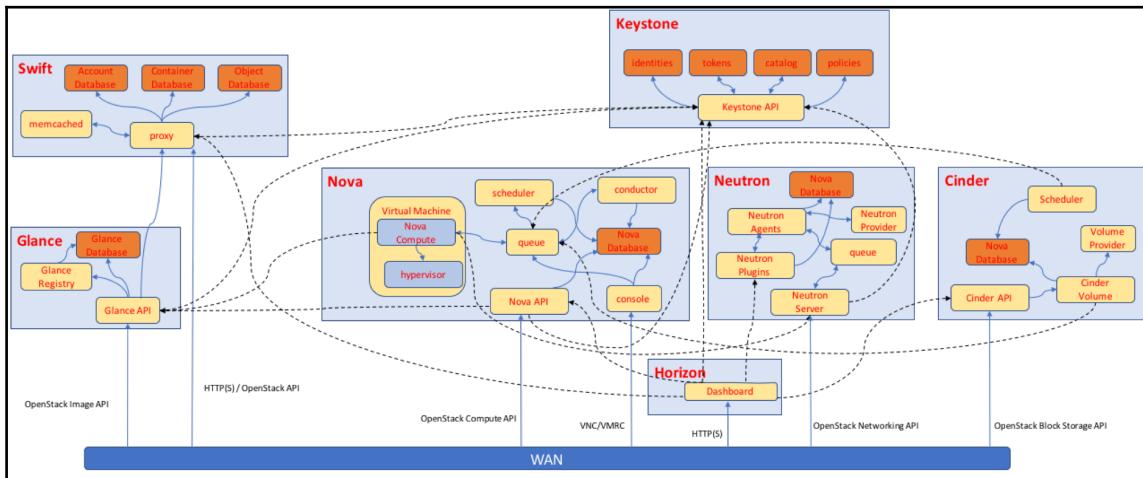
The hybrid architectural model is a combination of private and public clouds. Such combinations may be multiple public clouds used simultaneously or a combination of public and private cloud infrastructure. Organizations tend to favor a hybrid model if sensitive data needs unique management, while the frontend interface can make use of the reach and scale of the cloud. Another use case is maintaining a public cloud agreement to offset conditions where scalability exceeds the corporation's private cloud infrastructure. In this case, the public cloud will be used as a load balancer until the swell of data and usage drops back to the private cloud constraints. This use case is called cloud bursting and refers to the use of clouds as contingent resource provisions.

The OpenStack cloud architecture

OpenStack is an open source Apache 2.0 licensed framework used to build cloud platforms. It is primarily an IaaS and has been in the developer community since 2010. The OpenStack Foundation manages the software and has support from more than 500 companies, including Intel, IBM, Red Hat, and Ericsson. We will use OpenStack as a reference architecture for other cloud providers as much of the components and terminology are reused in commercial clouds.

OpenStack started as a joint project between NASA and Rackspace around 2010. The architecture has all the major components of other cloud systems, including compute and load balancing; storage components, including backup and recovery; networking components, dashboards, security and identity, data and analytics packages, deployment tools, monitors and meters, and application services. These are the components that an architect would look for when choosing a cloud service.

Architecturally, OpenStack is an interwoven layer of components. The basic form of an OpenStack cloud is shown in the following figure. Each service has a particular function and a unique name (such as Nova). The system acts as a whole, providing a scalable enterprise-class cloud functionality:



OpenStack top-level architectural diagram.

All communication within the OpenStack components is done through **Advanced Message Queueing Protocol (AMQP)** message queues, specifically the RabbitMQ or Qpid. Messages can either be non-blocking or blocking depending on how the message was sent. A message would be sent as a JSON object into RabbitMQ and receivers would find and fetch their messages from the same service. This is a loosely-coupled **remote procedure call (RPC)** method of communication between the major subsystems. The benefit in a cloud environment is that client and server vices are completely decoupled, and this allows the servers to dynamically scale up or down. Messages are not broadcast but directed, which keeps traffic to a minimum. You may also recall that AMQP is a common messaging protocol used in the IoT space.

Keystone – identity and service management

Keystone is the identity manager service of the OpenStack cloud. An identity manager establishes user credentials and login authorization. It is essentially the starting point or entry point into the cloud. This resource will maintain a central directory of users and their access rights. This is the top level of security to ensure user environments are mutually exclusive and secure. Keystone can interface with services like LDAP in an enterprise-level directory. Keystone also maintains a token database and delivers temporary tokens to users similarly to how **Amazon Web Services (AWS)** establishes credentials. A service registry is used to query what products or services are available to the user programmatically.

Glance – image service

Glance provides the heart of virtual machine management for OpenStack. Most cloud services will provide some degree of virtualization and will have an analog resource similar to Glance. The image service API is a RESTful service and lets a customer develop VM templates, discover available VMs, clone images to other servers, register VMs, and even move running virtual machines to different physical servers without interruption. Glance calls into Swift (the object store) to retrieve or store different images. Glance supports different styles of virtual images:

- raw: Unstructured images
- vhd: VMWare, Xen, Oracle VirtualBox
- vmdk: Common disk format
- vdi: QEMU emulator image
- iso: Optical drive image (CD ROM)
- aki/ari/ami: Amazon image



A virtual machine consists of the entire hard drive volume image content including guest operating systems, runtimes, applications, and services.

Nova compute

This is the heart of the OpenStack compute resource management service. Its purpose is to identify and appropriate compute resources based on demand. It also has the responsibility of controlling the system hypervisor and virtual machines. Nova will work with several VMs, as mentioned, such as VMware or Xen, or it can manage containers. On-demand scaling is part and parcel of any cloud offering.

Nova is based on a RESTful API web service for simplified control.

To get a list of servers one would `GET` the following into Nova through the API:

```
{your_compute_service_url}/servers
```

To create a bank of servers (ranging from a minimum of 10 and a maximum of 20) we would `POST` the following:

```
{
    "server": {
        "name": "IoT-Server-Array",
        "imageRef": "8a9a114e-71e1-aa7e-4181-92cc41c72721",
        "flavorRef": "1",
        "metadata": {
            "My Server Name": "IoT"
        },
        "return_reservation_id": "True",
        "min_count": "10",
        "max_count": "20"
    }
}
```

Nova would respond with a `reservation_id`:

```
{
    "reservation_id": "84.urcyplh"
}
```

Thus, the programming model is fairly simple in order to manage the infrastructure.

The Nova database is needed to maintain the current state of all objects in the cluster. For example, a few of the states the various servers in the cluster can include are:

- ACTIVE: Server is actively running
- BUILD: Server is in the process of being built and is not completed
- DELETED: Server has been deleted
- MIGRATING: Server is migrating to a new host

Nova relies on a scheduler to determine which task to execute and where to execute it. The scheduler can associate affinity randomly or use filters to choose a set of hosts that best match some sets of parameters. The filter end product will be an ordered list of host servers to use from best to worst (with incompatible hosts removed from the list).

The following is the default filter used to assign server affinity:

```
scheduler_available_filters = nova.scheduler.filters.all_filters
```

A custom filter can be created (for example, a Python or JSON filter called `IoTFilter.IoTFilter`) and attached to the scheduler as follows:

```
scheduler_available_filters = IoTFilter.IoTFilter
```

To set a filter to find servers that have 16 VCPUs programmatically through the API, we construct a JSON file as follows:

```
{
  "server": {
    "name": "IoT_16",
    "imageRef": "8a9a114e-71e1-aa7e-4181-92cc41c72721",
    "flavorRef": "1"
  },
  "os:scheduler_hints": {
    "query": "[>=,$vcpus_used,16]"
  }
}
```

Alternatively, OpenStack also allows you to control the cloud through a command-line interface:

```
$ openstack server create --image 8a9a114e-71e1-aa7e-4181-92cc41c72721 \
--flavor 1 --hint query='[>=,$vcpus_used,16]' IoT_16
```

OpenStack has a rich set of filters to allow for the custom allocation of servers and services. This allows for very explicit control of server provisioning and scaling. This is a classic and very important aspect of cloud design. Such filters include but are not limited to:

- RAM size
- Disk capacity and type
- IOPS levels
- CPU allocation
- Group affinities
- CIDR affinity

Swift – Object Storage

Swift provides a redundant storage system for the OpenStack data center. Swift allows clusters to scale by adding new servers. The object storage will contain things such as the accounts and containers. A user's virtual machine may be stored or cached in Swift. A Nova compute node can call directly into Swift and download the image on the first run.

Neutron – Networking services

Neutron is the OpenStack network management and VLAN service. The entire network is configurable and provides services such as:

- Domain name services
- DHCP
- Gateway functions
- VLAN management
- L2 connectivity
- SDN
- Overlay and tunneling protocols
- VPN
- NAT (SNAT and DNAT)
- Intrusion detection systems
- Load balancing
- Firewalls

Cinder – Block Storage

Cinder provides OpenStack with the persistent block storage services needed for a cloud. It acts as a *storage as a service* for use cases such as databases and dynamically growing file systems, including data lakes, which are of particular importance in streaming IoT scenarios. Like other components in OpenStack, the storage system is itself dynamic and scales as needed. The architecture is built on high availability and open standards.

The functionality provided by Cinder includes:

- Creation, deletion, and binding of storage devices to Nova compute instances
- Multiple storage vendor interoperability (HP 3PAR, EMC, IBM, Ceph, CloudByte, Scality)
- Support for multiple interfaces (Fibre Channel, NFS, Shared SAS, IBM GPFS, iSCSI)
- Backup and retrieval of disk images
- Snapshot images of points in time
- Alternative storage for VM images

Horizon

The final element covered here is Horizon. Horizon is the OpenStack dashboard. It is the single-pane-of-glass view into OpenStack for the customer. It provides a web-based view of the various components that comprise OpenStack (Nova, Cinder, Neutron, and others).

Horizon provides a user interface view of the cloud system as an alternative means over the API. Horizon is extensible so a third party can add their widgets or tools to the dashboard. A new billing component may be added and a Horizon dashboard element can then be instantiated for customers.

Most IoTs that use cloud deployments will include some form of a dashboard with similar features.

Heat – orchestration (optional)

Heat can launch multiple composite cloud applications and manage cloud infrastructure based on templates on an OpenStack instance. Heat integrates with telemetry to autoscale a system to match load needs. Templates in Heat try to comply with AWS CloudFormation formats, and relationships between resources can be specified in a similar manner (for example, this volume is connected to this server).

A Heat template may resemble the following:

```
heat_template_version: 2015-04-30

description: example template

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image }
      flavor: { get_param: flavor }
      admin_pass: { get_param: admin_pass }
      user_data:
        str_replace:
          template: |
            #!/bin/bash
            echo hello_world
```

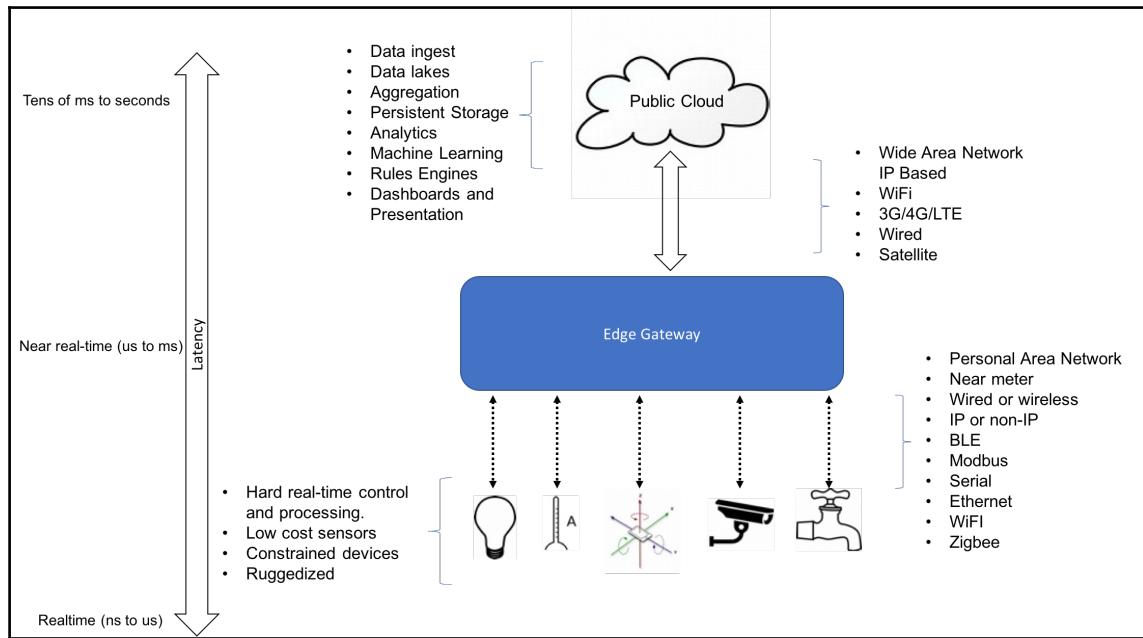
Ceilometer – telemetry (optional)

OpenStack provides an optional service called Ceilometer that can be used for telemetry data gathering and the metering of resources used by each service. Metering is used to collect information about usage and convert that into customer bills. Ceilometer also provides rating and billing tools. Rating is converting a billed value into equivalent currency, and billing is used to start a payment process.

Ceilometer monitors and meters different events, such as starting a service, attaching a volume, and stopping an instance. Metrics are gathered on CPU usage, the number of cores, memory usage, and data movement. All this is collected and stored in a MongoDB database.

Constraints of cloud architectures for IoT

A cloud service provider sits outside the IoT edge device and presides over the wide area network. One particular trait of the IoT architecture is that the PAN and WPAN devices may not be IP-compliant. Protocols such as **Bluetooth Low Energy (BLE)** and Zigbee are not IP-based while everything on the WAN including the cloud is IP-based. Thus, the role of the edge gateway is to perform that level of translation:



Latency effects in the cloud. Hard real-time response is critical in many IoT applications and forces processing to move closer to the endpoint device.

Latency effect

Another effect is the latency and response time for events. As you get closer to the sensor, you enter the realm of hard real-time requirements. These systems are typically deeply embedded systems or microcontrollers that have latency set by real-world events. For example, a video camera is sensitive to the frame rate (typically 30 or 60 fps) and must perform a number of sequential tasks in a data flow pipeline (demosaicing, denoising, white balance and gamma adjusting, gamut mapping, scaling, and compression). The amount of data flowing through a video imaging pipeline (1080p video using 8-bits per channel at 60 fps) is roughly 1.5 GB/s. Every frame must flow through this pipeline in real time, therefore most video image signal processors perform these transforms in silicon.

If we move up the stack, the gateway has the next best response time and is usually in the single digit millisecond range. The gating factor in the response time is the WPAN latency and the load on the gateway. As previously mentioned in the WPAN chapter, most WPANs such as BLE, are variable and dependent on the number of BLE devices under the gateway, scan intervals, advertisement intervals, and so on. BLE connection intervals can be as low as 7.5 ms but can vary depending on how the customer adjusts the advertisement intervals to minimize power usage. Wi-Fi signals typically have a 1.5 ms latency. Latency at this level requires a physical interface to the PAN. One wouldn't expect to pass raw BLE packets to the cloud with any hope of near real-time control.

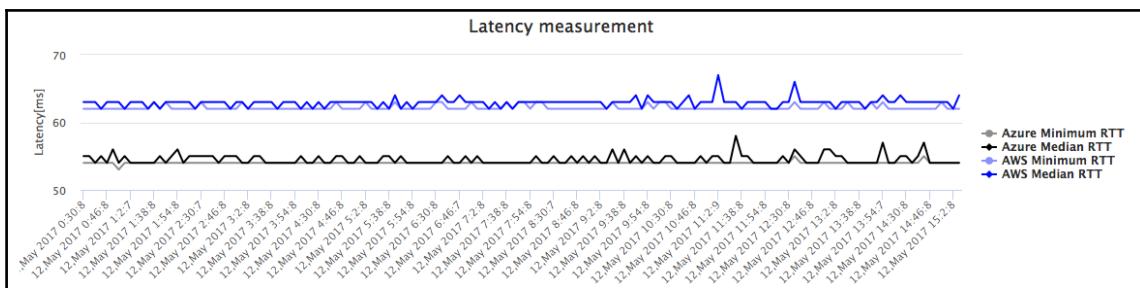
The cloud component introduces another degree of latency over the WAN. The route between the gateway and cloud provider can take multiple paths based on the locations of the data centers and the gateway. Cloud providers usually provide a set of regional data centers to normalize traffic. To understand the true latency impact of a cloud provider, one must sample the latency ping over the course of weeks or months and across regions:

Region	Latency
US East (Virginia)	80 ms
US East (Ohio)	87 ms
US West (California)	48 ms
US West (Oregon)	39 ms
Canada (Central)	75 ms
Europe (Ireland)	147 ms
Europe (London)	140 ms
Europe (Frankfurt)	152 ms

Asia Pacific (Mumbai)	307 ms
Asia Pacific (Seoul)	192 ms
Asia Pacific (Singapore)	306 ms
Asia Pacific (Sydney)	205 ms
Asia Pacific (Tokyo)	149 ms
South America (São Paulo)	334 ms
China (Beijing)	436 ms
GovCloud (US)	39 ms

Amazon AWS CloudPing from the US West Client courtesy of CloudPing.info (for more information, visit <http://www.cloudping.info>).

An exhaustive analysis of cloud latency and response times is kept by CLAudit (for more information, visit <http://claudit.feld.cvut.cz/index.php#>). Other tools also exist to analyze latency, such as Fathom and SmokePing (for more information visit <https://oss.oetiker.ch/smokeping/>). These sites research, monitor, and archive TCP, HTTP, and SQL database latency across AWS and Microsoft Azure on a daily basis across many regions worldwide. This produces the best visibility to the overall latency impact one may expect from a cloud solution. For example, the figure here illustrates the **round-trip times (RTT)** for a single day between a test client in the US communicating with Amazon AWS and Microsoft Azure servers in the Western US. It is also useful to note the variability in RTT. While a spike of 5 ms may be tolerable in many applications, it may lead to failure in a hard real-time control system or factory automation:



Round-trip times and latency for cloud providers. The chart illustrates latency in milliseconds for two cloud providers over several hours.

Typically, cloud latency will be in the order of tens if not hundreds of milliseconds without accounting for any overhead of processing for the incoming data. This should now set an expectation for the varying levels of response when building a cloud-based architecture for the IoT. Near-device architectures allow for sub-10 ms responses and also enjoy the benefit of being repeatable and deterministic. Cloud solutions can introduce variability into response times as well as an order of magnitude/greater response time than a near-edge device. An architect needs to consider where to deploy portions of the solution based on these two effects. Cloud providers should also be chosen based on their data center deployment models. If an IoT solution is being deployed worldwide or perhaps will grow to cover multiple regions, the cloud service should have data centers located in geographically similar areas to assist in normalizing the response time. The chart reveals the great variance in latency based on a single client reaching data centers worldwide. This is not an optimal architecture.

Fog computing

Fog computing is the evolutionary extension of cloud computing at the edge. This section details the difference between Fog and Edge computing and provides the various topologies and the architectural references for Fog Computing.

The Hadoop philosophy for Fog computing

Fog computing draws its analogy from the success of Hadoop and MapReduce, and to better understand the importance of Fog Computing, it is worth taking some time to think about how Hadoop works. MapReduce is a method of mapping and Hadoop is an open source framework based on the MapReduce algorithm.

MapReduce has three steps: map, shuffle, and reduce. In the **map** phase, computing functions are applied to local data. The shuffle step redistributes the data as needed. This is a critical step as the system attempts to collocate all dependent data to one node. The final step is the reduce phase, where the processing across all the nodes occurs in parallel.

The general takeaway here is that MapReduce attempts to bring processing to where the data is and not to move the data to where the processors are. This scheme effectively removes communication overhead and a natural bottleneck in systems that have extremely large structured or unstructured datasets. This paradigm applies to IoT as well. In the IoT space, data (possibly a very large amount of data) is produced in real time as a stream of data. This is the big data in IoT's case. It's not static data like a database or a Google storage cluster, but an endless live stream of data from every corner of the world. A fog-based design is the natural way to resolve this new big data problem.

Fog Computing versus Edge Computing versus cloud computing

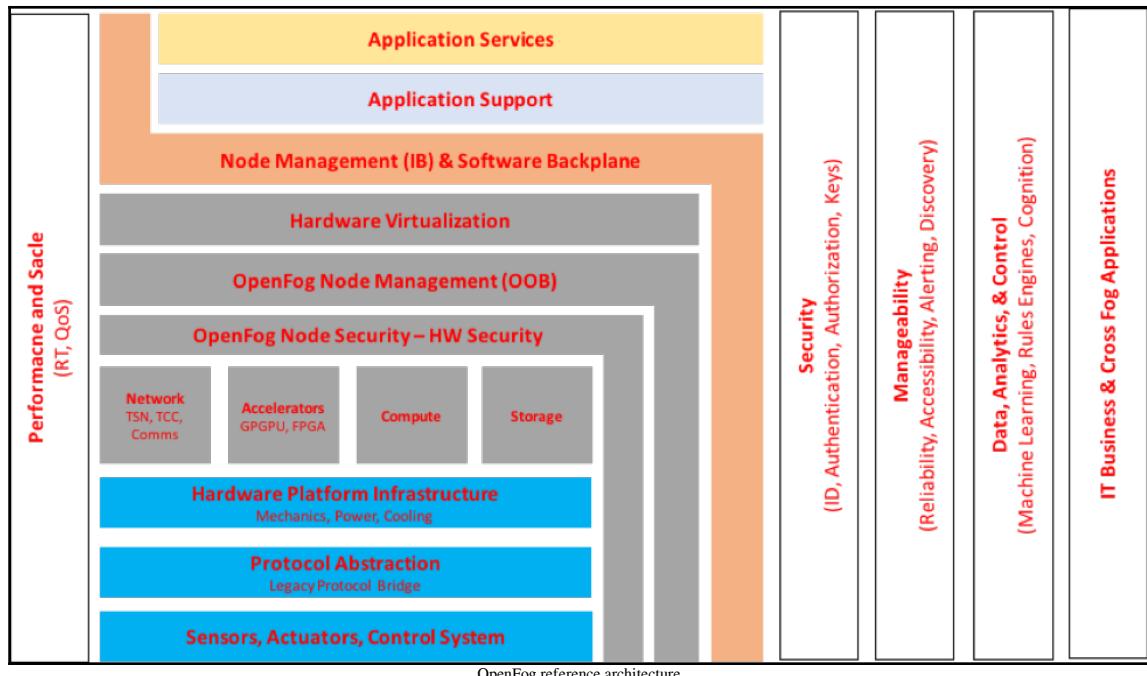
We have already defined edge computing as moving processing close to where data is being generated. In the case of IoT, an edge device could be the sensor itself with a small microcontroller or embedded system capable of WAN communication. Other times, the edge will be a gateway in architectures with particularly constrained endpoints hanging off the gateway. Edge processing is also usually referred to in a machine-to-machine context where there is a tight correlation between the edge (client) and a server located elsewhere. Edge computing exists, as stated, to resolve issues with latency and unnecessary bandwidth consumption, and to add services such as denaturing and security close to the data source. An edge device may have a relationship with a cloud service at the cost of latency and carrier; it doesn't actively participate in the cloud infrastructure.

Fog Computing is slightly different from an edge computing paradigm. Fog Computing first and foremost shares a framework API and communications standard with other fog nodes and/or an overlay cloud service. Fog nodes are extensions of the cloud whereas edge devices may or may not involve a cloud whatsoever. Another key tenet of Fog Computing is that the fog can exist in layers of hierarchy. Fog Computing can also load balance and steer data east-west and north-south to assist in resource balancing. From the previous section's definition of the cloud and the services it offers, one can think of these fog nodes as simply more (albeit less powerful) infrastructures in a hybrid cloud.

OpenFog Reference Architecture

A fog architectural framework such as a cloud framework is necessary to understand the interworking and data contracts between various layers. Here, we explore the OpenFog Consortium Reference Architecture: https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf. The OpenFog Consortium is a non-profit industry group chartered with defining the interoperability standards for Fog Computing. While not a standards body, they influence the direction of other organizations through liaison and industry influence. The OpenFog Reference Architecture is a model to assist architects and business leaders in creating hardware, building software, and acquiring infrastructure for Fog Computing. OpenFog realizes the benefit of cloud-based solutions and the desire to move that level of computing, storage, networking, and scaling to the edge without sacrificing latency and bandwidth.

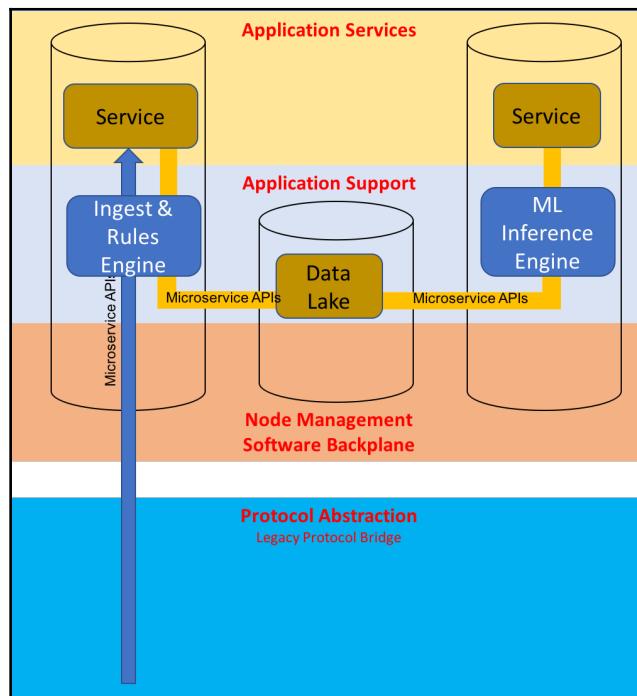
The OpenFog Reference Architecture consists of a layered approach from edge sensors and actuators at the bottom, to application services at the top. The architecture has some similarities with typical cloud architecture such as OpenStack, but extends this further since it is more analogous to a PaaS than an IaaS. For that matter, OpenFog provides a full stack and is generally hardware agnostic, or at least abstracts the platform from the rest of the system:



Application services

The role of the service layer is to provide the pane-of-glass and custom services needed for the mission. This includes providing connectors to other services, hosting data analytics packages, providing a user interface if needed, and provide core services.

The connectors in the application layer connect the services to the support layer. The protocol abstraction layer provides the pathway for a connector to talk directly to a sensor. Each service should be thought of as a micro-service in a container. The OpenFog Consortium advocates container deployment as the proper method for software to be deployed at the edge. This makes sense when we think of the edge devices as extensions of the cloud. An example of a container deployment could look like the following diagram. Each cylinder represents an individual container that can be deployed and managed separately. Each service then exposes APIs to reach between containers and layers:



Example OpenFog application. Here multiple containers can be deployed, each providing different services and support functions.

Application support

This is infrastructure componentry to help build the final customer solution. This layer may have dependencies in terms of how it was deployed (for example, as a container). Support comes in many forms, including:

- Application management (image identification, image verification, image deployment, authentication)
- Logging tools
- Registration of components and services
- Runtime engines (containers, VMs)
- Runtime languages (Node.js, Java, Python)
- Application servers (Tomcat)
- Message buses (RabbitMQ)
- Databases and archives (SQL, NoSQL, Cassandra)
- Analytic frameworks (Spark)
- Security services
- Web servers (Apache)
- Analytics tools (Spark, Drool)

OpenFog suggests these services should be packaged as containers, as shown in the preceding diagram. The reference architecture isn't a strict guideline and the architect must choose the correct amount of support that can be enabled on a constrained edge device. For example, the processing and resources may only allow for a simple rules engine and disallow anything like a stream processor, let alone a recurrent neural network.

Node management and software backplane

This refers to **In-Band (IB)** management and governs how a fog node communicates with other nodes in its domain. Nodes are also managed through this interface for upgrades, status, and deployment. The backplane can include the operating system of the node, custom drivers and firmware, communication protocols and management, file system control, virtualization software, and containerization for micro-services.

This level of the software stack touches nearly every other layer in the OpenFog Reference Architecture. Typical features of the backplane include:

- **Service discovery:** Allows for the ad hoc, fog-to-fog trust models.
- **Node discovery:** Allows for fog nodes to be added and for joins in a cluster similar to cloud- clustering techniques.
- **State management:** Allows for different computational models for stateful and stateless computes with multiple nodes.
- **Pub/Sub management:** Allows for data to be pushed rather than pulled. Additionally, it allows for levels of abstraction in software construction.

The OpenFog reference architecture, or for that matter any fog-based architecture, should allow for tiers of deployment. That is, a fog architecture isn't simply limited to a cloud connected to a fog gateway connected to a handful of sensors. In fact, there are multiple topologies dependent on the scale, bandwidth, processing load, and economies that can be designed. The reference architecture should provision itself for multiple topologies, just as the real cloud can dynamically scale and load balance itself based on demand.

Hardware virtualization

Like typical cloud systems, OpenFog defines the hardware as a virtualization layer. Applications should have no bindings to a particular set of hardware. Here the system should load balance across the fog and migrate or add resources as needed. All hardware components are virtualized at this level, including compute, network, and storage.

OpenFog node security

The consortium defines this level as the hardware security portion of the stack. Higher level fog nodes should be able to monitor lower-level fog nodes as part of the hierarchy in the topology (covered later). Peer nodes should be able to monitor their east-west neighbors. This layer also has the following responsibilities:

- Encryption
- Tamper and physical security monitors
- Packet inspection and monitoring (east-west and north-south)

Network

This is the first component of the hardware system layer. The network module is the east-west and north-south communication module. The network layer is cognizant of the fog topology and routing. It has the role of routing physically to other nodes. This is a major difference from a traditional cloud network that virtualizes all internal interfaces. Here the network has meaning and geographical presence in an IoT deployment. For example, a parent node hosting four other child nodes all attached to cameras may have the responsibility of aggregating video data from the four sources and sticking (fusing) the image content together to create a 360° field of view. To do this, it must know which child node pertains to which direction, and it cannot do that arbitrarily or randomly.

The network component requirements include:

- Resilient, in case a communication link goes down. In effect, it may need to understand how to rebuild a mesh to keep data flowing.
- The network layer is also the place for data translation and re-packaging from non-IP sensors to IP protocols. Examples of this are Bluetooth, Z-Wave, and wired sensors.
- Handling failover cases.
- Binding to a variety of communication fabrics (Wi-Fi, wired, 5G).
- Providing the typical network infrastructure needed for an enterprise deployment (security, routing, and so on).

Accelerators

Another aspect of OpenFog that differs from other cloud schemas is the notion of accelerator services. Accelerators are commonplace now in the form of GPGPUs and even FPGA to provide services for imaging, machine learning, computer vision and perception, signal processing, and encryption/decryption. OpenFog envisions fog nodes to be able to be resourced and allocated on an as-needed basis. One can force a second or third-level node farm in the hierarchy that could provide additional computing facilities as needed dynamically.

We can even force other forms of acceleration into the fog, for example:

- Nodes dedicated to bulk mass storage in the event a large data lake needs to be generated
- Nodes that include alternative communication links such as satellite radio in the catastrophic event that all land-based communication is lost

Compute

The compute portion of the stack is similar to the compute functionality of the Nova layer in OpenStack. The principal functions include:

- Task execution
- Resource monitoring and provisioning
- Load balancing
- Capability queries

Storage

The storage slice of the architecture maintains the low-level interface to the fog storage. The types of storage we spoke of earlier, such as data lakes or workspace memory, may be needed at the edge for hard real-time analysis. The storage layer will also manage all the traditional types of storage devices, such as:

- RAM arrays
- Rotating disks
- Flash
- RAID
- Data encryption

Hardware platform infrastructure

The infrastructure layer isn't so much an actual layer between software and hardware, but more of the physical and mechanical structure of the fog node. As fog devices will be in harsh and remote locations, they must be rugged and resilient as well as self-reliant. OpenFog defines the cases that need to be considered in a fog deployment, including:

- Size, power, and weight characteristics
- Cooling systems
- Mechanical support and retention
- Serviceability mechanics
- Physical attack resistance and reporting

Protocol abstraction

The protocol abstraction layer binds the lowest elements of the IoT system (sensors) with other layers of the fog node, other fog nodes, and the cloud. OpenFog advocates an abstraction model to identify and communicate with a sensor device through the protocol abstraction layer. By abstracting the interface to sensors and edge devices, a heterogeneous mixture of sensors can be deployed on a single fog node, for example, analog devices that pass through digital-to-analog converters as well as digital sensors. Even the interfaces to the sensors can be individualized, such as Bluetooth to temperature devices in vehicles, can bus to other engine sensors, SPI interfaced sensors on various vehicle electronics, and GPIO sensors to the various door and theft sensors. By abstracting the interface, the upper-layers of the software stack can access such disparate devices through a standardized approach.

Sensors, actuators, and control systems

This is the bottom end of the IoT stack: the actual sensors and devices at the edge. These devices can be smart, dumb, wired, wireless, near-range, far-field, and more. The association, however, is that they are communicating in some manner to the fog node and the fog node has the responsibility to provision, secure, and manage that sensor.

Amazon Greengrass and Lambda

In this section, we cover an alternative fog service called **Amazon Greengrass**. Amazon has provided years of world-class leading cloud services and infrastructure such as AWSs, S3, EC2, Glacier, and others. Since 2016, Amazon has invested in a new style of edge computing called **Greengrass**. Greengrass is an extension of AWS that allows a programmer to download a client to the fog, gateway, or smart sensor device.

Similarly, other fog frameworks, the intent is to provide a solution to reduce latency and response time, recuse bandwidth costs, and provide security for the edge. Features of Greengrass include:

- Cache data in case connection is lost
- Syncing data and device state to AWS cloud on reconnect
- Local security (authentication and authorization services)
- Message broker on the device and outside of the device
- Data filtering
- Command and control of device and data
- Data aggregation
- Operation while offline
- Iterative learning
- Calling any AWS service directly from Greengrass on the edge

To use Greengrass, a program will design a cloud platform in AWS IoT and define certain Lambda functions in the cloud. These Lambda functions are then assigned to edge devices, and deployed to those devices running the client, and authorized to execute Greengrass Lambdas. Currently, the Lambda functions are written in Python 2.7. Shadows are JSON abstractions in Greengrass that represent the state of the device and Lambda functions. These are synced back to AWS when desired.

Behind the scenes, communication between Greengrass on the edge and AWS in the cloud is done through MQTT.



Note that Lambda functions are not to be confused with Lambda architectures mentioned earlier. A Lambda function in the Greengrass context refers to an event-driven compute function.

An example Lambda definition used in Greengrass would look like the following. From the console in AWS we run a command line; we run the following tool and specify the Lambda function definition by name:

```
aws greengrass create-function-definition --name "sensorDefinition"
```

This will output the following:

```
{  
    "LastUpdatedTimestamp": "2017-07-08T20:16:31.101Z",  
    "CreationTimestamp": "2017-07-08T20:16:31.101Z",  
    "Id": "26309147-58a1-490e-a1a6-0d4894d6ca1e",  
    "Arn": "arn:aws:greengrass:us-  
west-2:123451234510:/greengrass/definition/functions/26309147-58a1-490e-  
a1a6-0d4894d6ca1e",  
    "Name": "sensorDefinition"  
}
```

We now create a JSON object with the Lambda function definitions and use the ID provided above and call `create-functiondefinition-version` from the command line:

- Executable is the Lambda function by name
- MemorySize is the amount of memory to allocate to the handler
- Timeout time in seconds before timeout counter expires

The following is an example of the JSON object to use with a Lambda function:

```
aws greengrass create-function-definition-version --function-definition-id  
"26309147-58a1-490e-a1a6-0d4894d6ca1e". --functions  
'[  
{  
    "Id": "26309147-58a1-490e-a1a6-0d4894d6ca1e",  
    "FunctionArn": "arn:aws:greengrass:us-  
west-2:123451234510:/greengrass/definition/functions/26309147-58a1-490e-  
a1a6-0d4894d6ca1e",  
    "FunctionConfiguration": {  
        "Executable": "sensorLambda.sensor_handler",  
        "MemorySize": 32000,  
        "Timeout": 3  
    }  
}]'
```

Several other steps are necessary to provision and create a subscription between the edge node and the cloud, but the Lambda handler will be deployed. This provides an alternative view of Fog Computing, as provided by Amazon. One can think of this model as a method of extending cloud services to an edge node and the edge being empowered to call on any resource provided by the cloud. This is by definition a true Fog Computing platform.

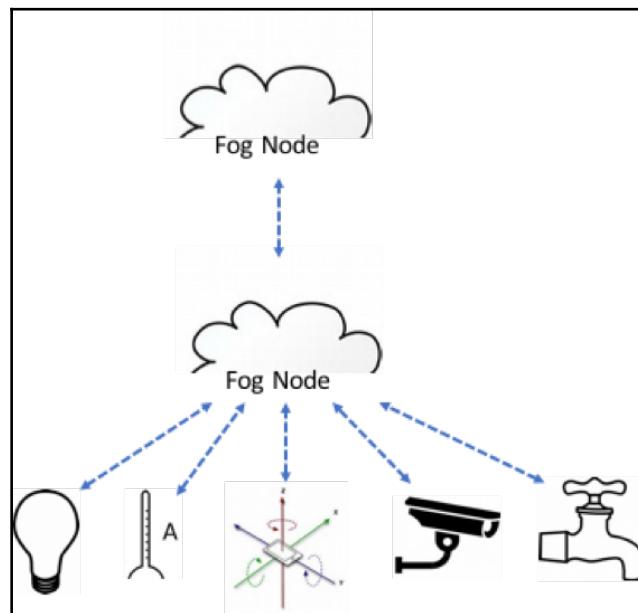
Fog Topologies

Fog topologies can exist in many forms, and the architect needs to consider several aspects when designing an end-to-end fog system. In particular, constraints such as cost, processing load, manufacturer interface, and east-west trafficking all come into play when designing the topology. A fog network can be as simple as a fog-enabled edge router connecting sensors to a cloud service. It can also grow in complexity to a multi-tier fog hierarchy with different degrees of processing ability and roles at each tier simultaneously distributing processing loads when and where necessary (east-west and north-south). Determining factors of the models are based on:

- **Data volume reduction:** For example, is the system collecting unstructured video data from thousands of sensors or cameras, aggregating the data, and looking for particular events in real time? if so then the dataset reduction will be significant as thousands of cameras will be producing hundreds of GBs of data daily, and fog nodes will need to distill large amounts of data into simple yes, no, danger, safe event tokens.
- **The number of edge devices:** If the IoT system is simply one sensor, then the dataset is small and may not justify a fog edge node at all. However, if the number of sensors grows or, in the worst case, the number of sensors is unpredictable and dynamic, then the fog topology may need to scale up or down dynamically. A use case would be a stadium venue using Bluetooth beaconing. As the audience grows for certain venues, the system must be able to scale non-linearly. At other times, the stadium may only seat a fraction of space and only need marginal processing and connectivity resources.
- **Fog node capabilities:** Depending on the topology structure and cost, some nodes may be better suited for connectivity to WPAN systems, while other nodes in the hierarchy may have additional processing capabilities for machine learning, pattern recognition, or image processing. An example would be edge fog nodes that manage a secure Zigbee mesh network and have special hardware for failover situations or WPAN security. Above that fog level would exist a fog processing node that would have extra RAM and GPGPU hardware to support the processing of raw data streaming from the WPAN gateways.

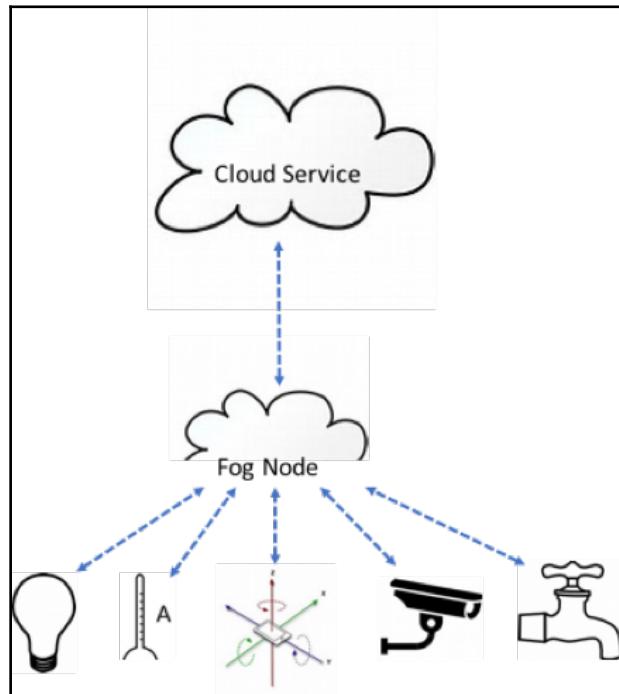
- **System reliability:** An architect may need to consider forms of failure in an IoT model. If one edge fog node were to fail, another could take its place to perform some action or service. This case is important in life-critical or real-time environments. In the same manner, extra fog nodes may be provisioned on demand; redundant nodes may be needed in fault-tolerant situations. In the case that there are no additional redundant nodes, some processing may be shared with neighbor nodes at the cost of system resources and latency, but the system will keep functioning. The final use case is where neighbor nodes act as watchdogs for each other. In the event a fog node fails or communication to the node fails, the watchdog will signal a failure event to the cloud and may perform some life-critical actions locally. A good example is in the event that a fog node fails to monitor traffic on a highway; a neighbor node may see the point failure, alert the cloud of the event, and signal on a billboard on the highway to reduce speed.

The simplest fog solution is an edge processing unit (gateway, thin clients, router) placed in proximity to a sensor array. Here, a fog node may be used as a gateway to a WPAN network or mesh and communicate to a host:



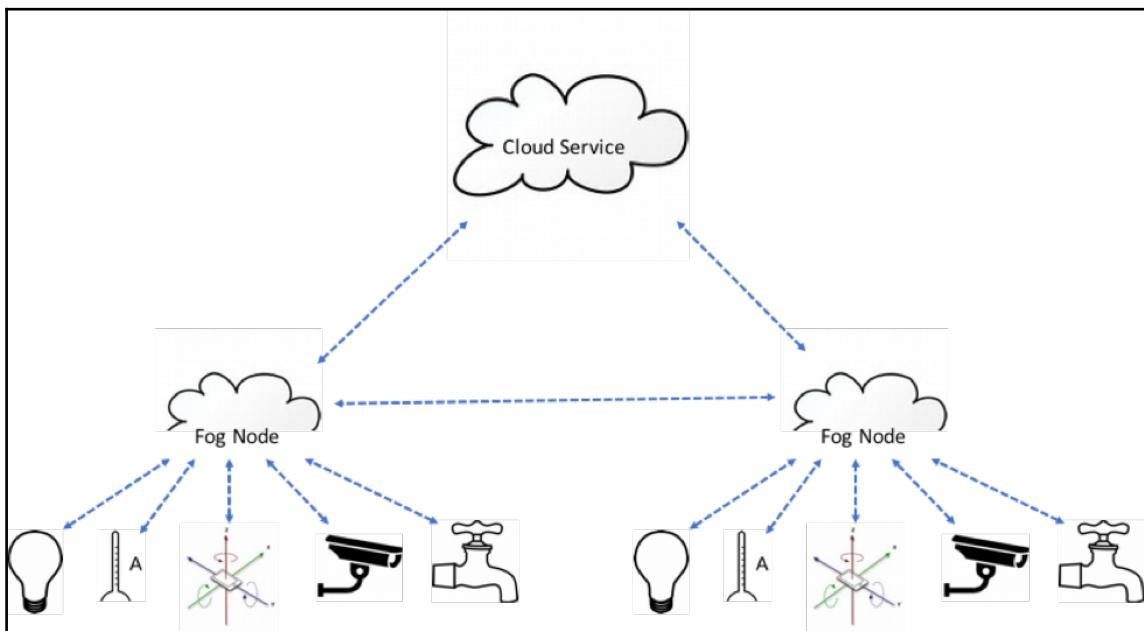
Simple fog topology. Edge-fog device manage an array of sensors and may communicate in a M2M manner with another fog node.

The next basic fog topology includes the cloud as the parent over a fog network. The fog node, in this case, will aggregate data, secure the edge, and perform the processing necessary to communicate with the cloud. What separates this model from edge computing is that the service and software layers of the fog node share a relationship with the cloud framework:



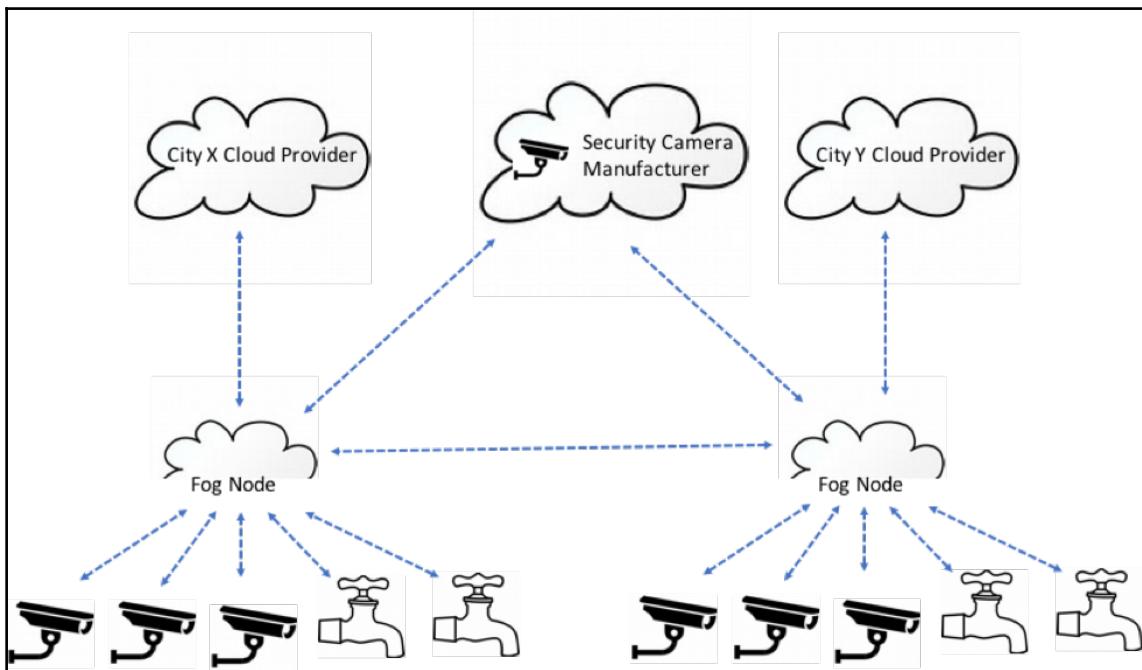
Fog to cloud topology. Here a fog node establishes a link to a cloud provider

The next model uses multiple fog nodes responsible for services and edge processing, and each connects to a set of sensors. A parent cloud provisions each fog node as it would a single node. Each node has a unique identity so as to provide a unique set of services based on geography. For example, each fog node may be at a different location for a retail franchise. Fog nodes may also communicate and traffic data east-west between edge nodes. An example use case would be a cold storage environment where a number of coolers and freezers need to be maintained and governed to prevent food spoilage. A retailer may have multiple coolers in multiple locations, all managed by a single cloud service but working with fog nodes at the edge:



Multiple fog nodes with a single master-cloud

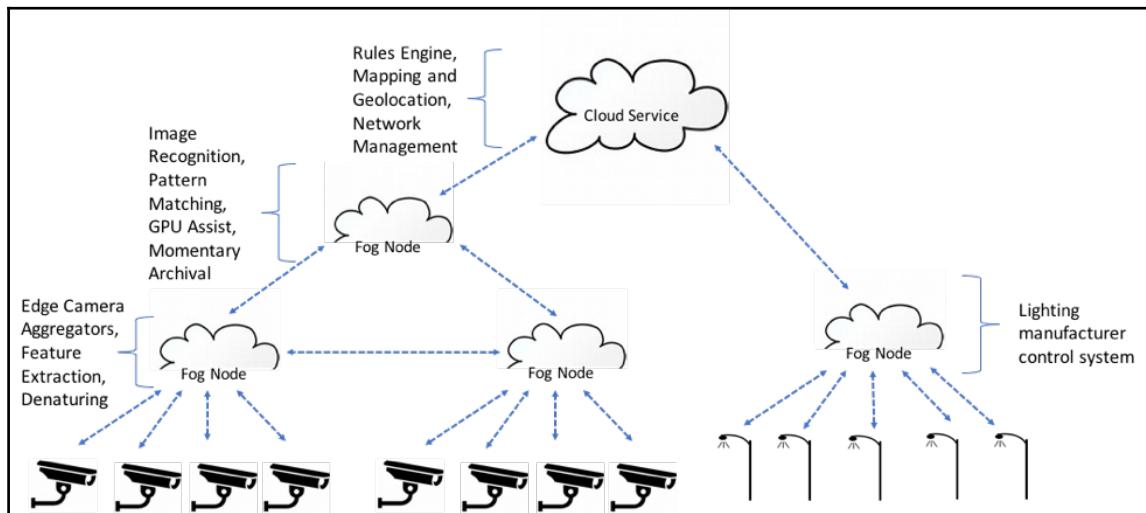
The next model extends topology two with the ability to communicate securely and privately to multiple cloud vendors from multiple fog nodes. In this model, multiple parent clouds may be deployed. For example, in smart cities, multiple geographical areas may exist and be covered by different municipalities. Each municipality may prefer one cloud provider over the other, but all municipalities are governed to use one approved and budgeted camera and sensor manufacturer. In that case, the camera and sensor manufacturer would have their single cloud instance coexist with multiple municipalities. The fog nodes must be able to steer data to multiple cloud providers:



Multiple fog nodes with multiple cloud providers. Clouds could be a mixture of public and private clouds.

Fog nodes also do not need a strict one-to-one relationship bridging sensors to clouds. Fog nodes can be stacked, tiered, or even kept in stasis until needed. Tiering layers of fog nodes above each other may sound counterintuitive if we are trying to reduce latency, but as mentioned previously, nodes can be specialized. For example, nodes closer to the sensors may provide hard real-time services or have cost constraints requiring them to have the minimal amount of storage and compute. A tier above them may provide the compute resources needed for aggregate storage, machine learning, or image recognition through the use of additional mass storage devices or GPGPU processors. The following example illustrates a use case in a city lighting example.

Here, a number of cameras sense moving pedestrians and traffic; the fog nodes closest to the cameras perform aggregation and feature extraction and pass those features upstream to the next tier. The parent fog node retrieves the features and performs the necessary image recognition through a deep learning algorithm. If an event of interest is seen (such as a pedestrian walking at night along a path), the event will be reported to the cloud. The cloud component will register the event and signal to a set of street lights in the pedestrian's vicinity to increase illumination. This pattern will continue as long as the fog nodes see the pedestrian moving. The end goal is the overall energy saved by not illuminating every street lamp to full intensity at all times:



Multi-tier fog topology. Fog nodes stack in a tier hierarchy to provide additional services or abstractions.

Summary

Analyzing data and deriving meaningful conclusions from a sensor is the goal of IoT. When we scale to thousands to millions and potentially billions of objects communicating and streaming data non-stop, we have to introduce advanced tools to ingest, store, marshal, analyze, and predict meaning from this sea of data. Cloud computing is one element in enabling that service in the form of clusters of scalable hardware and software. Fog Computing brings cloud processing closer to the edge to resolve issues with latency, security, and communication costs. Both technologies work together to run analytics packages in the form of rules engines to complex event processing agents. Choosing the model of cloud providers, frameworks, fog nodes, and analytics modules is a significant task and much literature goes deep into the semantics of programming and building these services. An architect must understand the topology and the end goal of the system to build a structure that meets today's needs and scales into the future.

In the next chapter, we will discuss the data analytics portion of IoT. The cloud certainly can host a number of analytic functions, however, we need to be prepared to understand that certain analysis should be performed on the edge close to the data source (sensor) or, if it makes more sense, in the cloud (using long-term historical data).

11

Data Analytics and Machine Learning in the Cloud and in the Fog

The value of an IoT system is not a single sensor event, or a million sensor events archived away. A significant value of IoT is in the interpretation and decision made of that data. While a world of billions of things connected and communicating with each other and the cloud is well and good, the value lies in what is within the data, what is not in the data, and what the patterns of data tell us. This is the data science and data analytics portions of IoT, and probably the most valuable area for the customer.

Analytics for the IoT segment deal with:

- Structured data (SQL storage), a predictable format of data.
- Unstructured data (raw video data or signals), a high degree of randomness and variance.
- Semi-structured (Twitter feeds), some degree of variance and randomness in form.

Data also may need to be interpreted and analyzed in real time as a streaming dataflow, or it may be archived and retrieved for deep analytics in the cloud. This is the data ingest phase. Depending on the use case, the data may need to be correlated with other sources in flight. In other cases, the data is simply logged and dumped to a data lake like a Hadoop database.

Next comes some type of staging, meaning a messaging system like Kafka will route data to a stream processor, or a batch processor, or perhaps both. Stream processing tolerates a continuous stream of data. Processing is typically constrained and very fast, as the data is processed in-memory. Therefore, processing must be as fast, or faster, than the rate of data entering the system. While stream processing provides near real-time processing in the cloud, when we consider industrial machinery and self-driving cars, stream processing does not provide hard real-time operating characteristics.

Batch processing, on the other hand, is efficient in dealing with high-volume data. It is particularly useful when IoT data needs to correlate against historical data.

After this phase, there may be a prediction and response phase where information may be presented on some form of the dashboard, logged, or perhaps the system will respond back to the edge device, where corrective actions can be applied to resolve some issue.

This chapter will discuss various data analysis models from complex event processing to machine learning. Several use cases will be taught to help generalize where one model can work and others may fail.

Basic data analytics in IoT

Data analytics intend to find events, usually in a streaming series of data. There are multiple types of events and roles that a real-time streaming analysis machine must provide. The following is a superset of analytic functions based on the work of Srinath Perera and Sriskandarajah Suhothayan, "Solution patterns for real-time streaming analytics." In Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems (DEBS '15). ACM, New York, NY, USA, 247-255. Following is an enumerated listing of these analytic functions:

- **Preprocessing:** Filter out events of little interest, denaturing, feature extraction, segmentation, transform data to a more suitable form (although data lakes prefer no immediate transformation), adding attributes to data such as a tag (data lakes do need tags).
- **Alerting:** Inspect data; if it exceeds some boundary condition, then raise an alert. The simplest example is if the temperature rises above a set limit on a sensor.

- **Windowing:** A sliding window of events is created that only draws rules upon that window. Windows can be based on time (for example, one hour), or length (2000 sensor samples). They can be sliding windows (for example, inspect only the 10 latest sensor events and produce a result whenever a new event arises), or batch windows (for example, produce an event only at the end of the window). Windowing is good for rules and for counting events. One could look for the number of temperature spikes in the last hour and resolve that a defect will occur on some machine.
- **Joins:** Combine multiple data streams into a new single stream. A scenario where this applies is a logistics example. Say a shipping company tracks their shipments with assets tracking beacons and that their fleet of trucks, planes, and facilities have geolocation information streaming as well. There are initially two streams of data: one for the package, and one for a given truck. When a truck picks up a package, those two streams become joined.
- **Errors:** Millions of sensors will generate missing data, garbled data, and data that is out of sequence. This is important in the IoT case with multiple streams of asynchronous and independent data. For example, data may be lost in a cellular WAN if a vehicle enters an underground parking garage. This analytic pattern correlates data within its own stream to attempt to find these error conditions.
- **Databases:** The analytics package will need to interact with some data warehouse. For example, if data is streaming in from a number of sensors looking, or in particular, when Bluetooth asset tags if an item is stolen or lost, a database of missing tag IDs would be referenced from all the gateways streaming in tag IDs to the system.
- **Temporal events and patterns:** This is most often used with the window pattern mentioned previously. Here, a series or sequence of events constitutes a pattern of interest. One can think of this as a state machine. Say we are monitoring the health of a machine based on temperature, vibrations, and noise. A temporal event sequence could be as follows:
 1. Detect if the temperature exceeds 100° C
 2. Then detect if vibrations exceed 1 m/s
 3. Next, detect if the machine is emitting noise at 110 dB
 4. If those events take place in that sequence, only then raise an alert
- **Tracking:** Tracking involves when or where something exists, an event occurred, or when something doesn't exist where it should have. A very basic example is geolocation of service trucks where a company may need to know exactly where a truck is, and when it was last there. This has application in agriculture, human movement, tracking patients, tracking high-value assets, luggage systems, smart city garbage, snow removal, and so on.

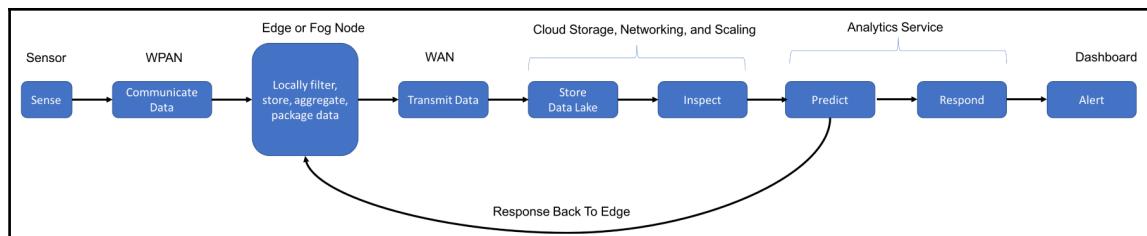
- **Trends:** This pattern is particularly useful for predictive maintenance. Here, a rule is designed to detect an event based on time-correlated series data. This is similar to temporal events, but differs in the sense that temporal events have no notion of time, only sequence order. This model uses time as a dimension in the process. A running history of time-correlated data could be used to find patterns like a livestock sensor in farming. Here, a head of cattle may wear a sensor that detects the animal movement and temperature. An event sequence can be constructed to see if the cattle moved in the last day. If there was no movement, the cattle may be sick or dead.
- **Batch queries:** Batch processing typically is more comprehensive and deeper than real-time stream processing. A well-designed streaming platform can fork analysis and call into a batch processing system. This will be talked about later in the form of Lambda processing.
- **Deep analytics pathway:** In real-time processing, we make a decision on the fly that some event has occurred. Whether or not that event really should signal an alarm may require further processing that will not operate in real time. This works because these events should be rare, and pass down information to a detailed analysis engine, while new events streaming in real time should be designed within a system. An example is a video surveillance system. Say a smart city issues an amber alert for a lost child. The smart city can issue a simple feature extraction and classification model for the real-time streaming engines. The model would detect license plates for a vehicle the child may be in, or potentially a logo on the child's shirt. The first step would be to image-capture license plate numbers of vehicles or logos on pedestrians, and send them to the cloud. The analysis package may identify a plate of interest or a logo out of millions of image samples as a first-level pass. That positively identified frame (and surrounding video frames) would be passed to a deeper analytics package that resolves the image with deeper object recognition algorithms (image fusion, super-resolution, machine learning) to eliminate false positives.
- **Models and training:** The first-level model described previously may, in fact, be an inference engine for a machine learning system. These machine learning tools are built on trained models that can be used for in-flight, real-time analysis.
- **Signaling:** It is often the case that an action needs to propagate back to the edge and sensor. A typical case is factory automation and safety. For example, if the temperature rises beyond a certain limit on a machine, log the event, but also send a signal back to the edge device to slow the machine down. The system must be able to be bidirectional in communication.

- **Control:** Finally, we need a way to control these analysis tools. Whether that is starting, stopping, reporting, logging, or debugging, facilities need to be in place to manage this system.

Now, we will concentrate on how to build a cloud-based analytics architecture that must ingest unpredictable and unstoppable streams of data, and deliver interpretations of that data as close to real-time as possible.

Top-level cloud pipeline

The following diagram is a typical flow of data from a sensor to a dashboard. Data will transit through several mediums (WPAN links, broadband, cloud storage in the form of a data lake, and so on). When we consider the following architectures to build a cloud analytics solution, we have to consider the effects of scaling. Choices made early in the design that are suitable to 10 IoT nodes and a single cloud cluster may not scale effectively when the number of endpoint IoT devices grows to the thousands, and are based in multiple geographies:



Typical IoT pipeline from sensor to cloud

The analytics (predict-respond) portion of the cloud can take on several forms:

- **Rules engines:** These simply define an action and produce an outcome.
- **Stream processing:** These are where events like sensor readings are injected into the stream processor. The processing path is a graph where nodes in the graph represent operators, and send events to other operators. The nodes contain the code for that portion of the processing, and a path to connect to the next node in the graph. This graph can be replicated and executed in parallel on a cluster so it is amenable to scaling up to hundreds of machines.
- **Complex event processing:** This is based on queries like SQL and written in that higher level language. It is based on event processing, and is tuned for low latency.

- **Lambda architecture:** This model attempts to balance throughput and latency by performing batch processing and stream processing in parallel on massive data sets.

The reason we talk about real-time analytics is that the data is streaming non-stop from millions of nodes simultaneously and asynchronously with various errors, format issues, and timings. New York City has 250,000 street lights (<http://www.nyc.gov/html/dot/html/infrastructure/streetlights.shtml>). Say each light is smart, meaning it monitors whether there is movement nearby, and if so, it brightens the light; otherwise, it remains dimmed to save power (2 bytes). Each light may also check if there is a problem with the light that needs maintenance (1 byte). Additionally, each light is monitoring temperature (1 byte) and humidity (1 byte) to help generate micro-climate weather predictions. Finally, the data also contains the light ID and a timestamp (8 bytes). The aggregate of all the lights nominally produces 250,000 messages a second, and can peak to 325,000 due to periods of rush hour, crowds, tourist sites, holidays, and so on. All in all, say our cloud service can process 250,000 messages per second, that implies a backlog of up to 75,000 events/second. If rush hour is truly 1 hour, then we backlog 270,000,000 events/hour. Only if we provide more processing in the cluster or reduce the incoming stream will the system ever catch up. If the incoming stream drops to 200,000 messages/sec during a quiet time, the cloud cluster will take 1.1 hours to resolve and consume 585 MB of memory (270 million backlogged messages at 13 bytes per message).

The following equations formalize the capacity of a cloud system:

$$C = \text{Cluster capacity} \left(\frac{\text{events}}{\text{s}} \right)$$

R_{event} = Event rate

T_{burst} = Time of burst of events

T_c = Time to complete backlog

$M_{backlog}$ = Message backlog (size)

$$\text{Backlog} = \begin{cases} 0 & \text{where } R_{event} \leq C \\ R_{event} - C & \text{where } R_{event} > C \end{cases}$$

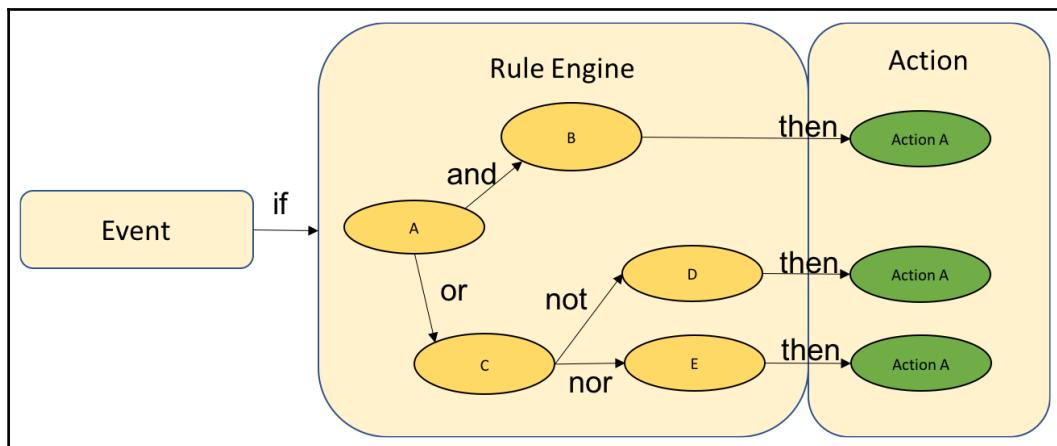
$$M_{backlog} = \text{Backlog} \times M_{size}$$

$$T_c = \frac{(R_{event} \times T_{burst}) + M_{backlog}}{C}$$

Rules engines

A rule engine is simply a software construct that executes actions on events. For example, if the humidity in a room exceeds 50%, send an SMS message to the owner. These are also called **Business Rule Management Systems (BRMS)**.

Rules engines may or may not have state and be called **stateful**. That is, it may have a history of the event, and take different actions depending on the order, the amount, or the patterns of events as they occurred historically. Alternatively, they may not maintain state, and only inspect the current event (stateless):



Simple rule engine example.

In our example of a rules engine, we will look at Drools. Drools is a BRMS developed by Red Hat and licensed under the Apache 2.0 license. JBoss Enterprise is a production version of the software. All objects of interest reside in the Drools *working memory*. Think of the working memory as the set of IoT sensor events of interest to compare to satisfy a given rule. Drools can support two forms of chaining: forward and backward. Chaining is a method of inference taken from game theory.

Forward chaining takes in available data until a rule chain is satisfied. For example, a rule chain may be a series of if/then clauses, as shown in the preceding diagram. Forward chaining will continuously search to satisfy one of the if/then paths to infer from an action. Backward chaining is the converse. Rather than starting with the data to be inferred from, we start with the action and work backward. The following pseudocode demonstrates a simple rule engine:

```
Smoke_Sensor = Smoke_Detected
Heat_Sensor = Heat_Detected

if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == Heat_Detected) then
    Fire
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor == Heat_Detected) then
    Furnace_On
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor == !Heat_Detected) then
    Smoking
if (Fire) then Alarm
if (Furnace_On) then Log_Temperature
if (Smoking) then SMS_No_Smoking_Allowed
```

Let us assume that:

- Smoke_Sensor: Off
- Heat_Sensor: On

Forward chaining would resolve the antecedent of the second clause, and infer that temperatures are being logged.

Backward chaining tries to prove that the furnace is on, and works backward in a series of steps:

1. Can we prove the temperatures are being logged? Take a look at this code:

```
if (Furnace_On) then Log_Temperature
```

2. Since the temperatures are being logged, the antecedent (Furnace_On) becomes the new goal:

```
if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor ==
Heat_Detected) then Furnace_On
```

3. Since the furnace is proven to be on, the new antecedent comes in two parts: `Smoke_Sensor` and `Heat_Sensor`. The rules engine now breaks it up into two goals:

```
Smoke_Sensor off  
Heat_Sensor on
```

4. The rules engine now attempts to satisfy both the subgoals. Upon doing so, the inference is complete.

Forward chaining has the advantage of responding to new data as it arrives, which can trigger new inferences.

Drools' semantic language is intentionally simple. Drools is composed of the following basic elements:

- Sessions, which define the default rules
- Entry points, which define the rules to use
- When statements, the conditional clause
- Then statements, the action to take

A basic Drools rule is shown in the following psuedo code. The `insert` operation places a modification in the working memory. You normally make a change to working memory when a rule evaluates to true.

```
rule "Furnace_On"  
when  
Smoke_Sensor(value > 0) && Heat_Sensor(value > 0)  
then  
insert(Furnace_On())  
end
```

After all the rules in Drool execute, the program can query working memory to see which rules evaluated to true using syntax, like the following:

```
query "Check_Furnace_On"  
$result: Furnace_On()  
end
```

A rule has two patterns:

- **Syntactic:** The format of data, parity, hash, range of values.
- **Semantic:** Value must belong to a set in a list, the count of high-temperature values must not exceed 20 in 1 hour. Essentially, these are meaningful events.

Drool supports the creation of very complex and elaborate rules to the point that a database of rules may be needed to store them. The semantics of the language allows for patterns, range evaluation, salience, times when a rule is in effect, type matching, and work on collections of objects.

Ingestion – streaming, processing, and data lakes

An IoT device is usually associated with some sensor or a device whose purpose is to measure or monitor the physical world. It does so asynchronously with respect to the rest of the IoT technology stack. That is, a sensor is always attempting to broadcast data, whether or not a cloud or fog node is listening. This is important, because the value of a corporation is in the data. Even if most of the data produced is redundant, there is always the opportunity that a significant event can occur. This is the data stream.

The IoT stream from a sensor to a cloud is assumed to be:

- Constant and never-ending
- Asynchronous
- Unstructured, or structured
- As close to real-time as possible

We discussed the cloud latency problem earlier in the [Chapter 10, Cloud and Fog Topologies](#). We also learned about the need for fog computing to help resolve the latency issue, but even without fog computing nodes, efforts are taken to optimize the cloud architecture to support IoT real-time needs. To do this, clouds need to maintain a flow of data and keep it moving. Essentially, data moving from one service to another in the cloud must do so as a pipeline, without the need to poll for data. The alternative form of processing data is called **batch processing**. Most hardware architectures treat data flow the same way, moving data from one block to another, and the process of data arrival triggers the next function. Additionally, careful use of storage and filesystem access is critical to reducing overall latency.

For this reason, most streaming frameworks will support in-memory operations and avoid the cost of temporary storage to a mass filesystem altogether. Michael Stonebraker called out the importance of data streaming in this fashion, see Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. 2005. "The 8 Requirements of Real-time Stream Processing.", SIGMOD Rec. 34, 4 (December 2005), 42-47. A well-designed message queue assists with this pattern. To build a successful architecture in a cloud that scales from hundreds of nodes to millions needs consideration.

The data stream will also not be perfect. With hundreds to thousands of sensors streaming asynchronous data more often than not, data will be missing (sensor lost communication), data will be poorly formed (error in transmission), or data will be out of sequence (data may flow to the cloud from multiple paths). At a minimum, a streaming system must:

- Scale with event growth and spikes
- Provide a publish/subscribe API to interface
- Approach near-real-time latency
- Provide scaling of processing of rules
- Support data lakes and data warehousing

Apache provides several open source software projects (under the Apache 2 license) that assist with building a stream processing architecture. Apache Spark is a stream processing framework that processes data in small batches. It is particularly useful when memory size is constrained on a cluster in the cloud (for example, < 1TB). Spark is built on in-memory processing, which has the advantages of reducing filesystem dependency and latency, as mentioned previously. The other advantage of working on batch data is that it is particularly useful when dealing with machine learning models, which will be covered later in this chapter. Several models, such as Convolutional Neural Network, can work on data in batches. An alternative from Apache is Storm. Storm attempts to process data as close to real-time as possible in a cloud architecture. It has a low-level API versus Spark and processes data as large events rather than dividing them up into batches. This has the effect of being low latency (sub-second performance).

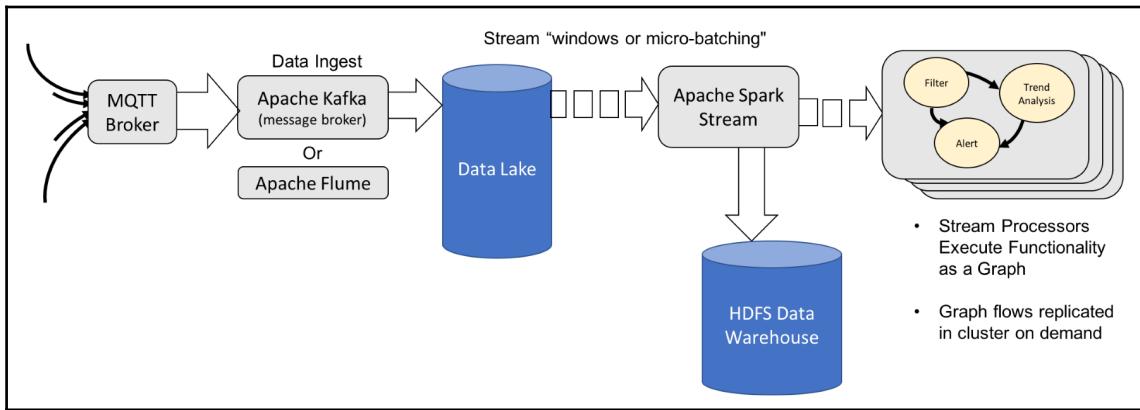
To feed the stream processing frameworks, we can use Apache Kafka or Flume. Apache Kafka is an MQTT on the ingest from various IoT sensors and clients, and connects to Spark or Storm on the outbound side. MQTT doesn't buffer data. If thousands of clients are communicating to the cloud over MQTT, some system will be needed to react to an incoming stream and provide the buffering needed. This allows Kafka to scale on demand (another important cloud attribute), and can react well to spikes in events. A stream of 1,00,000 events per second can be supported with Kafka. Flume, on the other hand, is a distributed system to collect, aggregate, and move data from one source to another, and is slightly easier to use *out-of-the-box*. It is also tightly integrated with Hadoop. Flume is slightly less scalable than Kafka, since adding more consumers means changing the Flume architecture. Both investors could stream in-memory without ever storing it; however, generally we don't want to do that, we want to take the raw sensor data and store it in as raw a form as possible with all the other sensors streaming in simultaneously.

When we think of IoT deployments in the thousands or millions of sensors and end nodes, a cloud environment may make use of a data lake. A data lake is essentially a massive storage facility holding raw unfiltered data from many sources. Data lakes are flat filesystems. A typical filesystem will be organized hierarchically, with volume, directories, files, and folders in a basic sense. A data lake organizes elements in its storage by attaching metadata element (tags) to each entry. The classic data lake model is Apache Hadoop, and nearly all cloud providers use some form of data lake underneath their services.

Data lake storage is particularly useful in the IoT, as it will store any form of data whether it is structured or unstructured. A data lake also assumes that all data is valuable, and will be kept permanently. This bulk persistent mass of data is optimal for the data analytic engines. Many of those algorithms function better based on how much data they are fed, or how much data is used to train their models.

A conceptual architecture using traditional batch processing and stream processing is illustrated in the following diagram. In the architecture, the data lake is fed by a Kafka instance. Kafka could provide the interface to Spark in batches and send data to a data warehouse.

There are several ways to reconfigure the topology in the following diagram, as the connectors between components are standardized:



Basic diagram of cloud ingestion engine to a data warehouse. Spark acts as the stream channel service.

Complex event processing

Complex event processing (CEP) is another analytic engine that is often used for pattern detection. From its roots in discrete event simulation and stock market volatility trading in the 1990s, it is by nature, a method capable of analyzing a live feed of streaming data in near real time. As hundreds and thousands of events enter the system, they are reduced and distilled into higher-level events. These are more abstract than raw sensor data. CEP engines have the advantage of a fast turnaround time in real-time analysis over a stream processor. A stream processor can resolve an event in the millisecond time frame. The downside is a CEP doesn't have the same level of redundancy, or dynamic scaling, as Apache Spark.

CEP systems use SQL-like queries, but rather than using a database backend, it searches an incoming stream for the pattern or rule you suggest. CEP consists of the tuple: discrete data element with a timestamp. A CEP makes use of the different analytic patterns described at the beginning of this chapter, and works well with a sliding window of events. Since it is SQL-like in semantics, and it is designed to be appreciably faster than a regular database query, all the rules and data reside in memory (usually a multi-GB database). Additionally, they need to be fed from a modern stream messaging system such as Kafka.

CEP has operations like sliding windows, joins, and sequence detection. Additionally, CEP engines can be based on forwarding or backward chaining as rules engines are. An industry-standard CEP system is the Apache WSO2 CEP. WSO2 coupled with Apache Storm can process over 1 million events per second, with no storage events needed. WSO2 is a CEP using an SQL language but can be scripted in JavaScript and Scala. The additional benefit is that it can be extended with a package called **Siddhi** to enable services such as:

- Geolocation
- Natural language processing
- Machine learning
- Time series correlation and regression
- Mathematical operations
- String and RegEx

Streams of data can be queried as in the following Siddhi QL code:

```
define stream SensorStream (time int, temperature single);
@name('Filter Query')
from SensorStream[temperature > 98.6]
select *
insert into FeverStream;
```

This all operates as discrete events allowing for sophisticated rules to be applied to millions of events transpiring simultaneously.

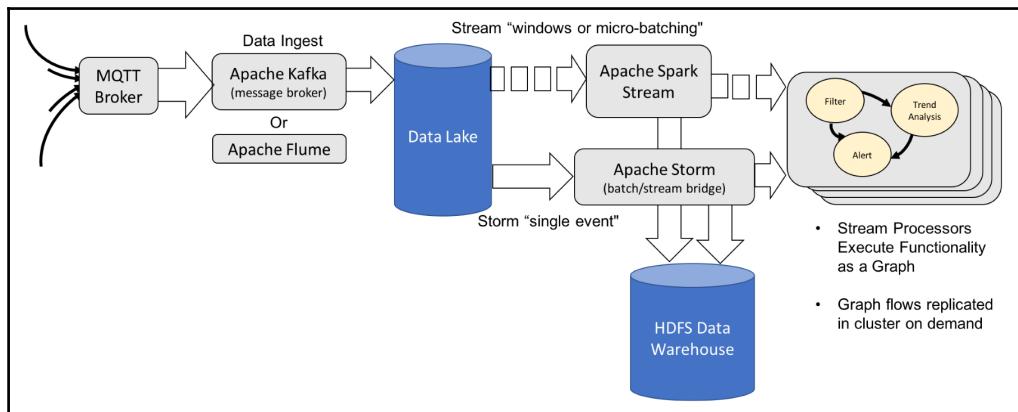
Having described a CEP, it is best for the architect to understand where a CEP and rules engine should be used. If the evaluation is a simple state, such as two ranges of temperatures, then the system is stateless, and a simple rules engine should be used. If the system maintains a temporal notion or a series of states, then a CEP should be used.

Lambda architecture

A Lambda architecture attempts to balance latency with throughput. Essentially, it mixes batch processing with stream processing. Similar to the general cloud topology of OpenStack or other cloud frameworks, Lambda ingests and stores to an immutable data repository. There are three layers of the topology:

- **Batch layer:** The batch layer is usually based on Hadoop clusters. The batch layer is significantly slower in processing than the stream layer. By sacrificing latency, it maximizes throughput and accuracy.

- **Speed layer:** This is the real-time in-memory data stream. The data can be erroneous, missing, and out of order. Apache Spark, as we have seen, is very good at providing a stream processing engine.
- **Service layer:** Service layer is where the recombination of batch and stream results are stored, analyzed, and visualized. Typical components of the service layer are: Druid, which provides facilities for combining batch and speed layers; Apache Cassandra for scalable database management; and Apache Hive, for data warehousing.



- Complexities of a Lambda Architecture. Here, a batch layer migrates data to the HDFS storage and the speed layer is delivered directly to a real-time analysis package via Spark.

Lambda architectures are, by nature, more complex than the other analytics engines. They are hybrid, and add additional complexity and resources to run successfully.

Sector use cases

We will now try to consider the typical use cases in a variety of industries adopting IoT and cloud analytics. When architecting the solution, we need to consider the scale, the bandwidth, real-time needs, and types of data to derive the correct cloud architecture, as well as the correct analytics architecture.

These are generalized examples - it is imperative to understand the entire flow and future scale/capacity when drawing a similar table:

Industry	Use cases	Cloud services	Typical bandwidth	Real time	Analytics
Manufacturing	<ul style="list-style-type: none"> Operational technology Brownfield Asset tracking Factory automation 	<ul style="list-style-type: none"> Dashboards Bulk storage Data lakes SDN (hybrid cloud topology) Low latency 	<ul style="list-style-type: none"> 500 GB/day/factory part produced 2 TB/minute mining operations 	Less than 1s	<ul style="list-style-type: none"> Recurrent neural nets Bayesian networks
Logistics and transport	<ul style="list-style-type: none"> Geolocation tracking Asset tracking Equipment sensing 	<ul style="list-style-type: none"> Dashboards Logging Storage 	<ul style="list-style-type: none"> Vehicles: 4 TB/day/vehicle (50 sensors) Aircraft: 2.5 to 10 TB/day (6000 sensors) Assets tracking: 1 MB/day/beacon 	<ul style="list-style-type: none"> Less than 1s (real-time) Daily (batch) 	Rule engines
Healthcare	<ul style="list-style-type: none"> Asset tracking Patient tracking Home health monitoring Wireless health equipment 	<ul style="list-style-type: none"> Reliability and HIPPA Private cloud option Storage and archival Load balancing 	<ul style="list-style-type: none"> 1 MB/day/sensor 	<ul style="list-style-type: none"> Less than 1s: Life critical Non-life critical: On each change 	<ul style="list-style-type: none"> Recurrent Neural Networks (RNN) Decision trees Rules engines
Agriculture	<ul style="list-style-type: none"> Livestock health and location tracking Soil chemistry analysis 	<ul style="list-style-type: none"> Bulk storage - archiving Cloud-to-cloud provisioning 	<ul style="list-style-type: none"> 512 KB/day/livestock head 1000 to 10000 head of cattle per feedlot 	<ul style="list-style-type: none"> 1 second (real-time) 10 minutes (batch) 	Rules engines
Energy	<ul style="list-style-type: none"> Smart meters Remote energy monitoring (solar, natural gas, oil) Failure prediction 	<ul style="list-style-type: none"> Dashboards Data lakes Bulk storage for Historical rate prediction SDN Low latency 	<ul style="list-style-type: none"> 100-200 GB/day/wind turbine 1 to 2 TB/day/oil rig 100 MB/day/smart meter 	<ul style="list-style-type: none"> Less than 1s: energy production 1 minute: smart meters 	<ul style="list-style-type: none"> RNN Bayesian networks Rules engines
Consumer	<ul style="list-style-type: none"> Real-time health logging Presence detection Lighting and heating/AC Security Connected home 	<ul style="list-style-type: none"> Dashboards PaaS Load balancing Bulk storage 	<ul style="list-style-type: none"> Security camera: 500 GB/day/camera Smart device: 1-1000 KB/day/sensor-device Smart home: 100 MB/day/home 	<ul style="list-style-type: none"> Video: less than 1s Smart home: 1s 	<ul style="list-style-type: none"> Convolutional neural nets (image sensing) Rules engines
Retail	<ul style="list-style-type: none"> Cold chain sensing POS machines Security systems Beaconing 	<ul style="list-style-type: none"> SDN/SDP Micro-segmentation Dashboards 	<ul style="list-style-type: none"> Security: 500 GB/day/camera General: 1-1000 MB/day/device 	<ul style="list-style-type: none"> POS and credit transaction: 100ms Beaconing: 1s 	<ul style="list-style-type: none"> Rules engines Convolutional neural networks for security
Smart City	<ul style="list-style-type: none"> Smart parking Smart trash pickup Environmental sensors 	<ul style="list-style-type: none"> Dashboards Data lakes Cloud-to-cloud services 	<ul style="list-style-type: none"> Energy monitors: 2.5 GB/day/city (70K sensors) Parking spots: 300 MB/day (80,000 sensors) Waste monitors: 350 MB/day (200,000 sensors) Noise monitors: 650 MB/day (30,000 sensors) 	<ul style="list-style-type: none"> Electric meters: 1 minute Temperature: 15 minutes Noise: 1 minute Waste: 10 minutes Parking spots: every change 	<ul style="list-style-type: none"> Rules engine Decision trees

Machine learning in IoT

Machine learning is not a new computer science development. On the contrary, mathematical models for data fitting and probability go back to the early 1800s, and Bayes' theorem and the least squares method of fitting data. Both are still widely used in machine learning models today, and we will briefly explore them in the chapter.

It wasn't until Marvin Minsky (MIT) produced the first neural network devices called **perceptrons** in the early 1950s that computing machines and learning were unified. He later wrote a paper in 1969 that was interpreted as a critique of the limitations of neural networks. Certainly, during that period, computational horsepower was at a premium. The mathematics were beyond the reasonable resources of IBM S/360 and CDC computers. As we will see, the 1960s introduced much of the mathematics and foundations of artificial intelligence in areas such as neural nets, support vector machines, fuzzy logic, and so on.

Evolutionary computation such as genetic algorithms and swarm intelligence became a research focus in the late 1960s and 1970s, with work from Rechenberg, Ingo Evolutionsstrategie (1973). It gained some traction in solving complex engineering problems. Genetic algorithms are still used today in mechanical engineering, and even automatic software design.

The mid-1960s also introduced the concept of hidden Markov models as a form of probabilistic AI, like Bayesian models. It had been applied to research in gesture recognition and bioinformatics.

Artificial intelligence research lulled with government funding drying up until the 1980s and the advent of logic systems. This started the field of AI known as a logic-based AI and supporting programming languages called Prolog and LISP, which allowed programmers to easily describe symbolic expressions. Researchers found limitations with this approach to AI: principally logic-based semantics didn't *think* like a human. Attempts using *anti-logic* or *scruffy* models to try to describe objects didn't work well either. Essentially, one cannot describe an object precisely using loosely coupled concepts. Later in the 1980s, expert systems took root. Expert systems are another form of logic-based systems for a well-defined problem trained by experts in that particular domain. One could think of them as a rule-based engine for a control system. Expert systems proved successful in corporate and business settings and became the first commercially available AI systems sold. New industries started to form around expert systems. These types of AI grew, and IBM used the concept to build *deep thought* to defeat chess grandmaster Garry Kasparov in 1997.

Fuzzy logic first manifested itself in research by Lotfi A. Zadeh at UC Berkeley in 1965, but it wasn't until 1985 that researchers at Hitachi demonstrated how fuzzy logic could be applied successfully to control systems. That sparked significant interest in Japanese automotive and electronics firms to adopt fuzzy systems into actual products. Fuzzy logic has been used successfully in control systems, and we will discuss it formally later in this chapter.

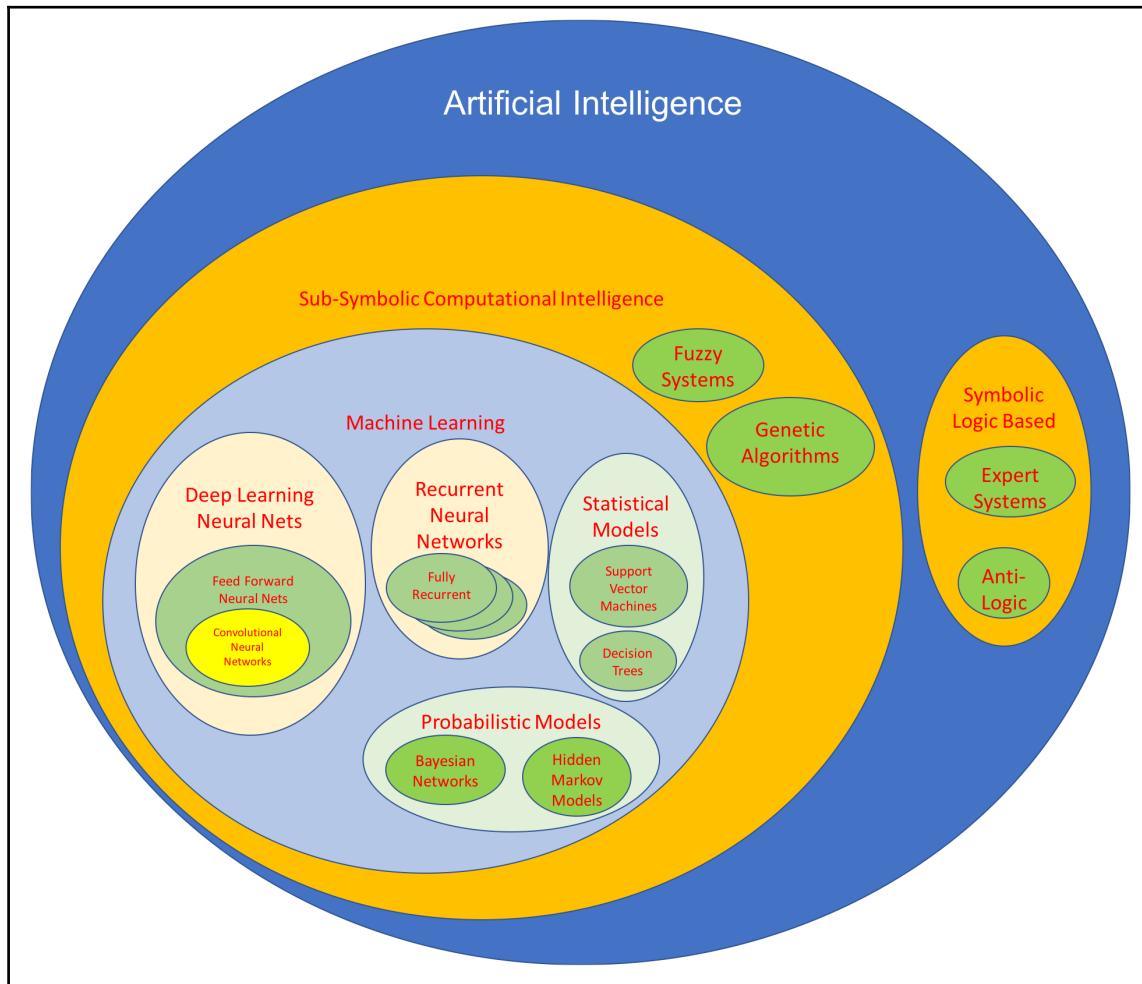
While expert systems and fuzzy logic seemed to be the mainstay for AI, there was a growing and noticeable gap between what it could do, and what it would never be able to do. Researchers in the early 1990s saw that expert systems, or logic-based systems, in general, could never emulate the mind. The 1990s brought on the advent of statistical AI in the form of hidden Markov models and Bayesian networks. Essentially, computer science adopted models commonly used in economics, trade, and operations research to make decisions.

Support vector machines were first proposed by Vladimir N. Vapnik and Alexey Chervonenkis in 1963, but became popular after the AI winter of the 1970s and early 1980s. SVMs became the foundation for linear and nonlinear classification by using a novel technique to find the best hyperplanes to categorize data sets. This technique became popular with handwriting analysis. Soon, this evolved into uses for neural networks.

RNN also became a topic of interest in the 1990s. This type of network was unique and different to deep learning neural networks such as Convolutional Neural Network, because it maintained state and could be applied to a problem involving the notion of time, such as audio and speech recognition. They have a direct impact on IoT predictive models today, which we will discuss later in this chapter.

A seminal event occurred in 2012 in the field of image recognition. In a competition, teams around the globe competed on a computer science task of recognizing the object in a 50-pixel by 30-pixel thumbnail. Once the object was labeled, the next task was to draw a box around it. The task was to do this for 1 million images. A team from the University of Toronto built the first deep Convolutional Neural Network to process images to win this competition. Other neural networks had attempted this machine vision exercise in the past, but the team developed an approach that identified images with more accuracy than any approach before, with an error rate of 16.4%. Google developed another neural net that brought the error rate down to 6.4%. It was also around this time that Alex Krizhevsky developed AlexNet, which introduced GPUs to the equation to greatly speed up training. All these models were built around Convolutional Neural Network, and had processing requirements that were prohibitive until the advent of GPUs.

Today, we find AI everywhere, from self-driving cars to speech recognition in Siri, to tools emulating humans in online customer service, to medical imaging, to retailers using machine learning models to identify consumer interest in shopping and fashion as they move about a store:



The Spectrum of Artificial Intelligence algorithms.

What does this have to do with IoT at all? Well, IoT opens up the spigot to a massive amount of constantly streaming data. The value of a system of sensors is not what one sensor measures, but what a collection of sensors measure and tell us about a much larger system. IoT, as mentioned earlier, will be the catalyst to generate a step-function in the amount of data collected. Some of that data will be structured: time-correlated series. Other data will be unstructured: cameras, synthetic sensors, audio, and analog signals. The customer wants to create useful decisions for their business based on that data. For example, in a manufacturing plant that is planning to optimize operational expenses and potentially capital expenses by adopting IoT and machine learning (at least, that's what they were sold on). When we think about a factory IoT use case, the manufacturer will have many interdependent systems. They may have some assembly tool to produce a widget, another robot to cut parts out of metal or plastic, another machine to perform some type of injection molding, conveyor belts, lighting and heating systems, packaging machine, supply and inventory control systems, robots to move material around, and various levels of controls systems. In fact, this company may have many of these spaces spread across a campus or geography. A factory like this has adopted all the traditional models of efficiency, and read W. Edwards Deming's literature; however, the next industrial revolution will come in the form of IoT and machine intelligence.

Specialized individuals know what to do when an erratic event occurs. For example, a technician who had been operating one of the assembly machines for years knows when the machine needs service based on how that machine is behaving. It may start creaking in a certain way. Perhaps it's worn out its ability to pick and place parts, and dropped a few in the last couple of days. These simple behavioral effects are things that machine learning can see and predict even before a human. Sensors can surround such devices, and monitor actions both perceived and inferred. An entire factory could be *perceived* in such a case to understand how that factory is performing at that very instant based on a collection of millions or billions of events from every machine and every worker in that system.

With that amount of data, only a machine learning appliance can sift through the noise and find the relevance. These are not human-manageable problems, but the manageable problem of big data and machine learning.

Machine learning models

We will now focus on specific machine learning models that have applicability to IoT. There is no single winner of the model one will use to sift through a collection of data. Each model has its particular strength and the use case it serves. The goal of any machine learning tool is to arrive at a prediction or inference of what a set of data is telling you. You want to be better than flipping a coin of 50%.

There are two types of learning systems to consider which are as follows:

- **Supervised learning:** It simply implies that the training data provided to the model has an associated label with each entry. For example, a set may be a collection of pictures each labeled with the content of that image: for example, cat, dog, banana, car. Many machine learning models today are supervised. Supervised learning allows for classification and regression problems to be solved. We will discuss classification and regression later on in this chapter.
- **Unsupervised learning:** It has no label for the training data. Obviously, this type of learning cannot resolve an image of a dog to the label *dog*. This type of learning model uses mathematical rules to reduce redundancy. A typical use case is to find clusters of like things.

There also exists a hybrid of both models, called **semi-supervised learning**, which mixes labeled data and unlabeled data. The goal is to force the machine learning model to organize data as well as make inferences.

The three fundamental uses of machine learning are:

- Classification
- Regression
- Anomaly detection

There are dozens of machine learning and AI constructs that could be talked about with application to IoT, but that would extend far beyond the scope of this book. We will concentrate on a small set of models to understand where they fit in relation to each other, what they target, and what their strengths are. We want to explore the uses and limitations of statistical, probabilistic, deep learning, and RNN, as they are the prevalent areas applicable for IoT artificial intelligence.

Within each of these large segments, we will generalize and dive into the following:

- **Random forests:** Statistical models (fast model, good for systems with many attributes needed for anomaly detection)
- **Bayesian networks:** Probabilistic models
- **Convolutional Neural Network:** Deep learning (deep learning model for unstructured image data)
- **RNN:** Recurrent neural nets (deep learning model for time series analysis)

Some models are not applicable anymore in the artificial intelligence space, at least for the IoT use cases we consider. So we will not focus on logic-based models, genetic algorithms, or fuzzy logic.

We will first talk through some initial nomenclature around classifiers and regression.

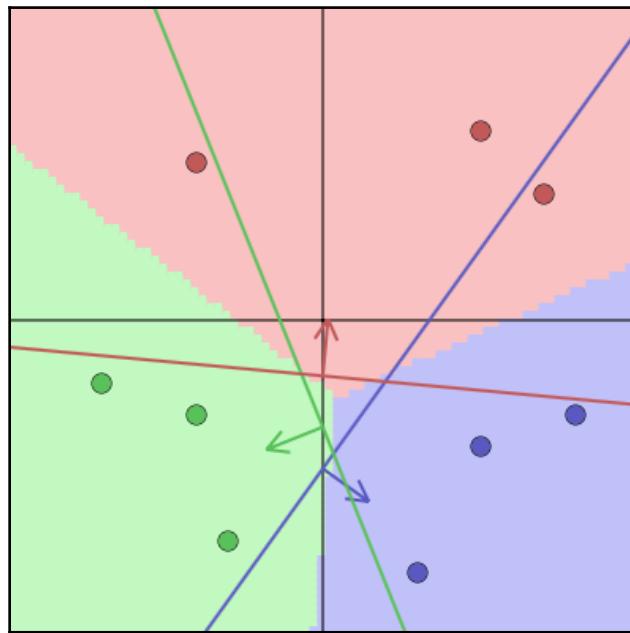
Classification

In most of the machine learning models, we will talk about how we start with classification. Classification is a form of supervised learning where the data is used to pick a name, value, or category. For example, using a neural network to scan images to find pictures of a shoe. In this field, there are two variants of classification:

- **Binomial:** If you are picking between one of two categories (coffee, tea)
- **Multi-class:** If there are more than two options

We use the Stanford linear classifier tool to help understand the concept of hyperplanes (<http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/>). The following diagram shows a trained learning system's attempts to find the best hyperplane to divide colored balls. We can see that after several thousand iterations, the division is somewhat optimal, but there still are issues with the top-right region, where the corresponding hyperplane includes a ball that belongs to the top hyperplane. Shown below is an example of a less than optimal classification.

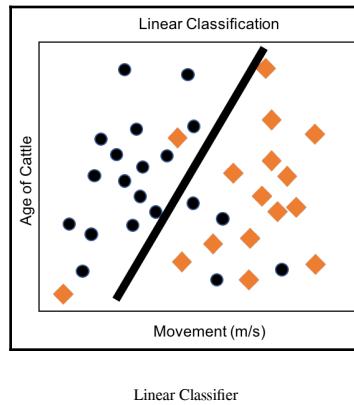
Here, hyperplanes are used to create artificial segments. The top right shows a single ball that should be classified with the other two balls at the top but was classified to belong to the bottom-right set.



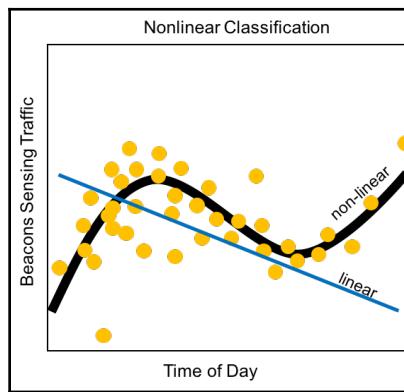
Less than optimal classification.

Notice in the preceding example from Stanford that the hyperplane is a straight line. This is called a **linear classifier** and includes such constructs as support vector machines (which attempts to maximize the linearity), and logistic regression (which can be used for binomial class and multi-class fitting). The following graph shows a binomial linear classification of two data sets: circles, and diamonds.

Here, a line attempts to form a hyperplane to divide two distinct regions of variables. Notice the best linear relationship does include errors:



Nonlinear relationships are also common in machine learning, and using a linear model would cause severe error rates. The following graph shows the linear curve fit versus the nonlinear. One issue with the nonlinear model is the tendency to overfit the test series. As we will see later, this has the propensity to make the machine learning tool accurate when executed on the training test data, but useless in the field. Following figure is a comparison of a linear versus nonlinear classifier:



Here, an n^{th} order polynomial curve attempts to build a much more precise model of the set of data points. Highly-precise models tend to fit a known training set well but have failed when presented with real-world data.

Regression

Classification is concerned with predicting a discrete value (circle or diamonds), whereas regression models are used to predict a continuous value. For example, regression analysis would be used to predict the average selling price for your home based on the selling prices of all the homes in your neighborhood and surrounding neighborhoods.

Several techniques exist to form regression analysis:

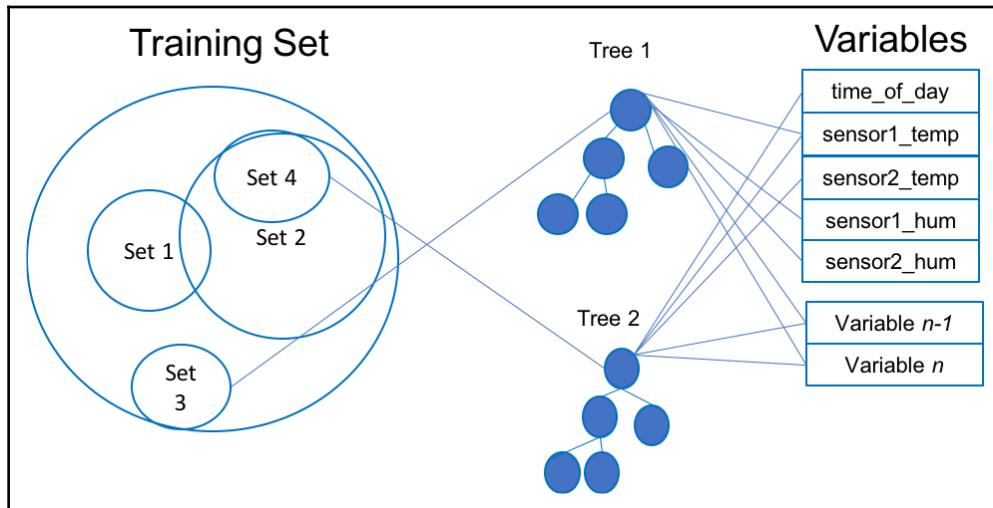
- Least squares method
- Linear regression
- Logistic regression

Random forest

A random forest is a subset of another machine learning model called the **decision tree**. A decision tree, as the diagram at the start of this section shows, is a group of learning algorithms that are part of the statistical set. A decision tree simply takes several variables into consideration and produces a single output that classifies the set. Each element evaluated is called the **set**. The decision tree produces a set of probabilities that a path has taken based on the input. One form of a decision tree is the **Classification and Regression Test (CART)**, developed by Breiman in 1983.

We now introduce the notion of bootstrap aggregating or *bagging*. When you have a single decision tree being trained, it is susceptible to noise injected into it and can form a bias. If, on the other hand, you have many decision trees being trained, we can lessen the chance of bias. Each tree will pick a random set of the training data or samples.

The output of the random forest training processes a decision tree based on a random selection of the training data, and a random selection of variables:



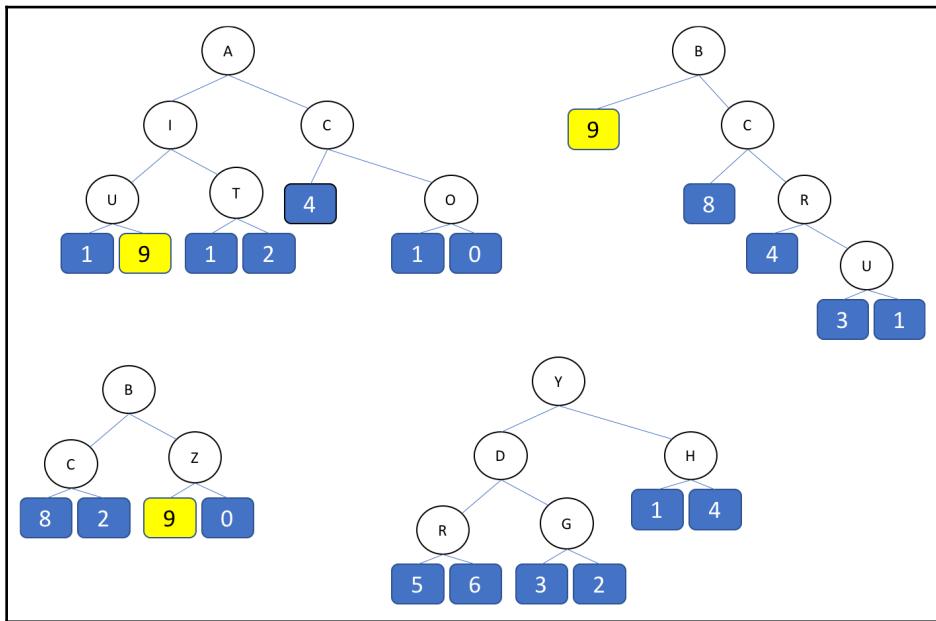
Random forest model. Here, two forests are constructed to pick a random, but not the whole set of variables.

Random forests extend bagging by not only selecting a random sample set, but also a subset of the number of features being qualified. This can be seen in the preceding image. This is counter-intuitive, since you want to train on as much data as possible. The rationale is that:

- Most trees are accurate, and provide correct predictions for most data
- Errors in the decision tree may happen at different places, in different trees

This is the rule of *group think* and *majority decisions*. If the outcome of several trees agree with each other even though they arrived at that decision through different paths and a single tree is an outlier, one will naturally side with the majority. This creates a model with low variance, compared to a single decision tree model that can be extremely biased. We can see the following example with four trees in a random forest. Each has been trained on a different subset of data, and have chosen random variables. The result of the flow is that three of the trees produce a result of 9, while the fourth tree produces a different result.

Regardless of what the fourth tree produced, the majority agreed by a different data set, different variable, and different tree structures that the result of the logic should be a 9:



Majority decision of a random forest. Here, several trees based on a random collection of variables, arrive at 9 as a decision. By arriving at a similar answer based on different input generally reinforces the model.

Bayesian models

Bayesian models are based on Bayes' theorem from 1812. Bayes' theorem describes the probability that an event will occur based on prior knowledge of the system. For example, what is the probability that a machine will fail based on the temperature of the device?

Bayes' theorem is expressed as:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

A and B are the events of interest. $P(A|B)$ asks, what is the probability that event A will occur, given event B has occurred? They have no relation to each other, and are mutually exclusive.

The equation can be re-written using the theorem of total probability, which replaces $P(B)$. We can also extend this to i number of events. $P(B|A)$ is the probability that event B will occur, given event A has occurred. This is the formal definition of Bayes' theorem:

$$P(A_i|B) = \frac{P(B|A_i) \times P(A_i)}{P(B|A_1) \times P(A_1) + P(B|A_2) \times P(A_2) + \dots + P(B|A_i) \times P(A_i)}$$

In the case, we are dealing with a single probability and its complement (pass/fail). The equation can be re-written as:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B|A) \times P(A) + P(B|A') \times P(A')}$$

An example follows. Supposing we have two machines producing identical parts for a widget. Say a machine can fail if its temperature exceeds a certain value. Machine A will fail 2% of the time if the temperature exceeds a certain temperature. Machine B will fail 4% of the time if it exceeds a certain temperature. Machine A produces 70% of the parts, and machine B produces the remaining 30%. If I pick up a random part and it fails, what is the probability it was produced by machine A , or machine B ?

In this case, A is an item produced by machine A , and B is an item produced by machine B . F represents the failed chosen part. We know:

- $P(A) = 0.7$
- $P(B) = 0.3$
- $P(F|A) = 0.02$
- $P(F|B) = 0.04$

Therefore, the probability you pick a defective part from machine A or B is:

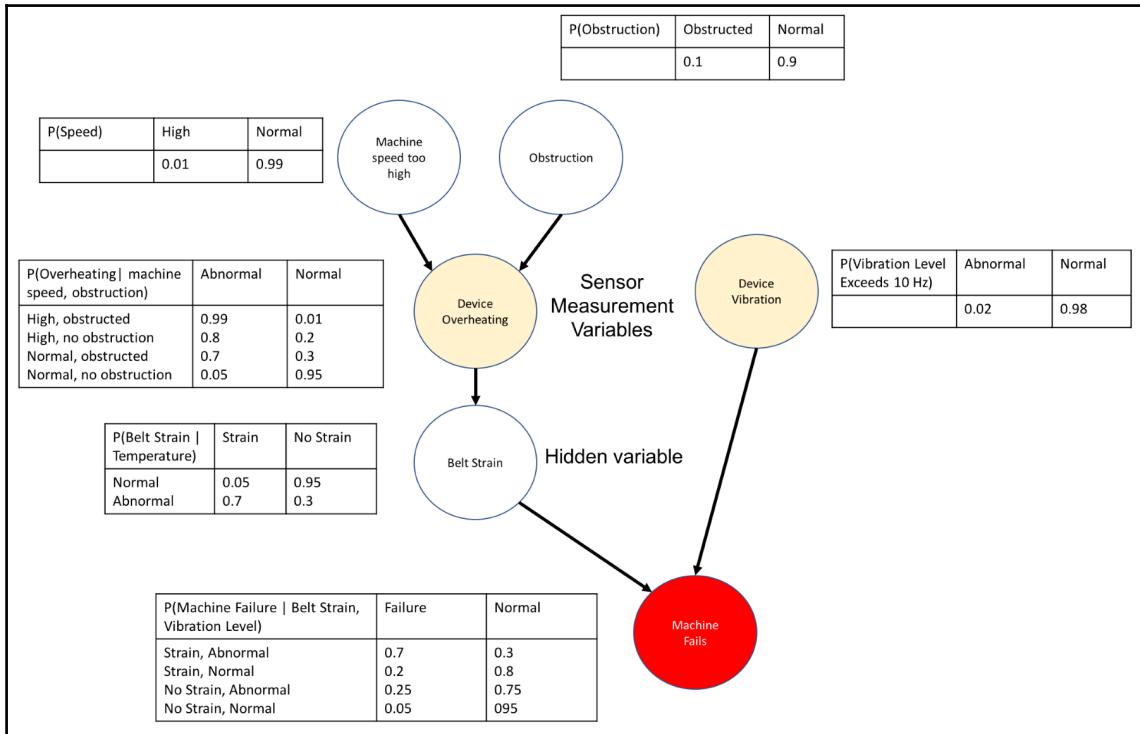
$$P(A | F) = \frac{P(F|A) \times P(A)}{P(F|A) \times P(A) + P(F|B) \times P(B)}$$

Replacing the values:

$$P(A | F) = \frac{0.02 \times 0.7}{(0.02 \times 0.7) + (0.04 \times 0.3)}$$

Therefore, $P(A | F) = 53\%$ and $P(B | F)$ is the complement $(1 - 0.53) = 47\%$.

A Bayesian network is an extension of Bayes' theorem in the form of a graphical probability model, specifically a directed acyclic graph. Notice the graph flows one way, and there are no loopbacks to previous states; this is a requirement of the Bayesian network:



Bayesian network model

Here, the various probabilities of each state come from expert knowledge, historical data, logs, trends, or combinations of these. This is the training process for a Bayesian network. These rules can be applied to a learning model in an IoT environment. As sensor data streams in, the model could predict machine failures. Additionally, the model could be used to make inferences. For example, if the sensors are reading an overheating condition, one could infer that there is a probability it may be related to the speed of the machine, or an obstruction.

There are variants of Bayesian networks that go beyond the scope of this book, but have benefits for certain types of data and problem sets:

- Naive Bayes
- Gaussian Naive Bayes
- Bayesian belief networks

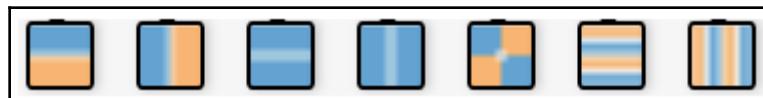
A Bayesian network is good for environments in IoT that can't be completely observed. Additionally, in a situation where the data is unreliable, Bayesian networks have an advantage. Poor sample data, noisy data, and missing data have less an effect on Bayesian networks than other forms of predictive analytics. The caveat is that the number of samples will need to be very large. Bayesian methods also avoid the overfitting problem, which we will discuss later in neural networks. Additionally, Bayesian models fit well with streaming data, which is a typical use case in IoT. Bayesian networks have been deployed to find aberrations in signals and time-correlated series from sensors, and also to find and filter malicious packets in networking.

Convolutional Neural Networks

A **Convolutional Neural Networks (CNN)** is a form of artificial neural networks in machine learning. We will first inspect the CNN, and then move on to the RNN. A CNN has proven to be very reliable and accurate at image classification, and is used in IoT deployments for visual recognition, especially in security systems. It is a good starting point to understand the process and mathematics behind any artificial neural network. Any data that can be represented as a fixed bitmap (say, a 1024 x 768 pixel image in three planes). A CNN attempts to classify an image to a label (for example, cat, dog, fish, bird) based on an additive set of decomposable features. The primitive features that compose image content are built up from small sets of horizontal lines, vertical lines, curves, shades, gradient directions, and so on.

First layer and filters

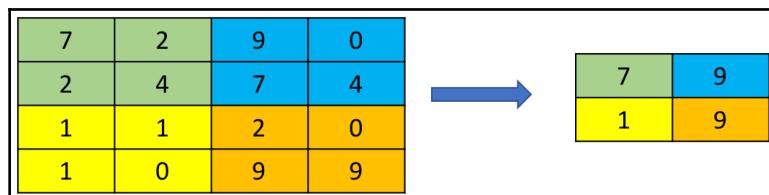
This basic set of features in the first layer of a CNN would be feature identifiers such as: small curves, tiny lines, color splotch, small distinguishing features (in the case of image classifiers). The filters will convolve around the image looking for similarities. The convolution algorithm will take the filter and multiply-sum the resultant matrix values. Filters activate when a specific feature results in a high-activation value:



First Layer of CNN. Here, large primitives are used to pattern match input.

Max pooling and subsampling

The next layer will typically be a pooling or max pooling layer. This layer will use input of the values from the last layer, and return the maximum value for a set of neighbor neurons, which is used as input to a single neuron in the next convolution layer. This is essentially a form of subsampling. Typically, the pooling layer will be a 2x2 subregion matrix as the result:



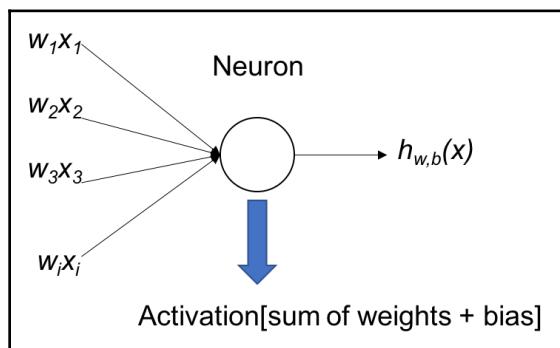
Max pooling. Attempts to find the maximum value in a sliding window across an image.

Pooling has several options: maximizing (as shown in the preceding diagram), averaging, and other sophisticated methods. The purpose of max pooling is to state that a particular feature was found within a region of the image. We don't need to know the exact position, just a general locality. This layer also recurs the dimensions we have to deal with, which ultimately affect the neural network performance, memory, and CPU usage. Max pooling also controls overfitting. Researchers have learned that if the neural network becomes finely tuned to images without this type of subsampling, it will work well on the training set of data it was programmed with, but will fail miserably with real-world images.

Hidden layers and formal description on forwarding propagation

The second convolutional layers use the results of the first layer as input. Remember, the input to the first layer was the original bitmap. The output of the first layer actually represents the location in the 2D bitmap, where a specific primitive was seen. The features of the second layer are more comprehensive than the first. The second layer would have composite structures such as splines and curves. Here, we will describe the role of the neuron, and the computations necessary to force an output from a neuron.

The role of the neuron is to input the sum of all the weights entering it against the pixel values. In the following graph, we see the neuron accepting inputs from the previous layer in the form of weights and bitmap values. The role of the neuron is to sum the weights and values and force them through an activation function as input to the next layer:

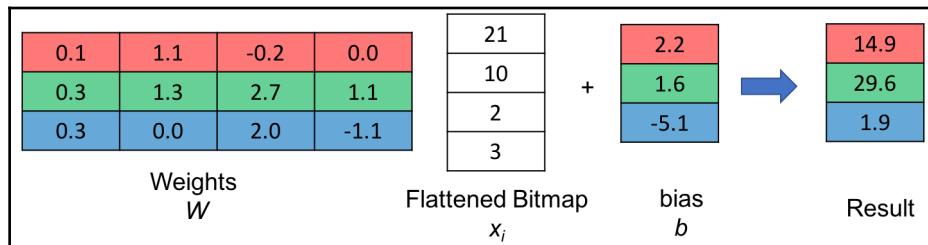


CNN basic element. Here, the neuron is a basic unit of computing with weights and other bitmap values taken as input. The neuron fires (or not) based on the activation function.

The equation for the neuron function is:

$$\sigma \left(\sum_i w_i x_i + b \right)$$

This can be a very large matrix multiplication problem. The input image is flattened into a one-dimensional array. The bias provides a method to influence the output without interacting with the real data. In the following diagram, we see an example of a weight matrix multiplied by a flattened one-dimensional image, and with the added bias. Note that in actual CNN devices you can add the bias to the weight matrix, and add a single 1.0 value to the bottom of the bitmap vector as a form of optimization. Here, the second value in the result matrix of 29.6 is the value chosen:



Matrix relationship of CNN. Here, the weights and bitmap are matrix multiplied and added to a bias.

The input values are multiplied by the weighting on each entering the neuron. This is a simple linear transform in matrix math. That value needs to pass through an activation function to determine if the neuron should fire or not. A digital system built on transistors takes voltages as input, and if the voltage meets a threshold value, the transistor switches on. The biological analog is the neuron that behaves nonlinearly to the input. Since we are modeling a neural network, we attempt to use nonlinear activation functions. Typical activation functions that can be chosen include:

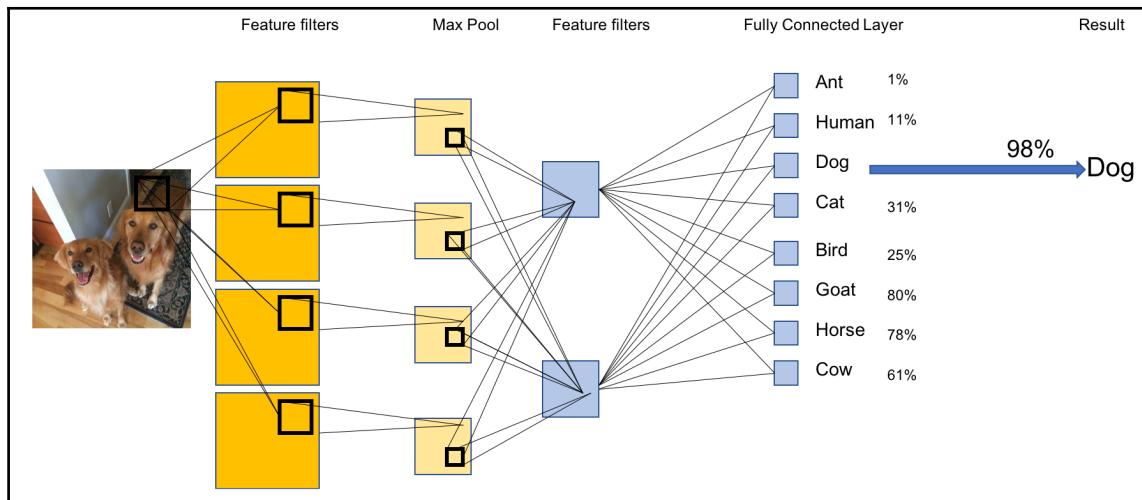
- Logistic (sigmoid)
- tanH
- **Rectified linear unit (ReLU)**
- **Exponential Linear Unit (ELU)**
- sinusoidal

The sigmoid activation function is:

$$\sigma(x) = \frac{1}{(1 + e^{-x})}$$

Without a sigmoid (or any type of activation function) layer, the system would be a linear transformation function and have substantially less accuracy for image or pattern recognition.

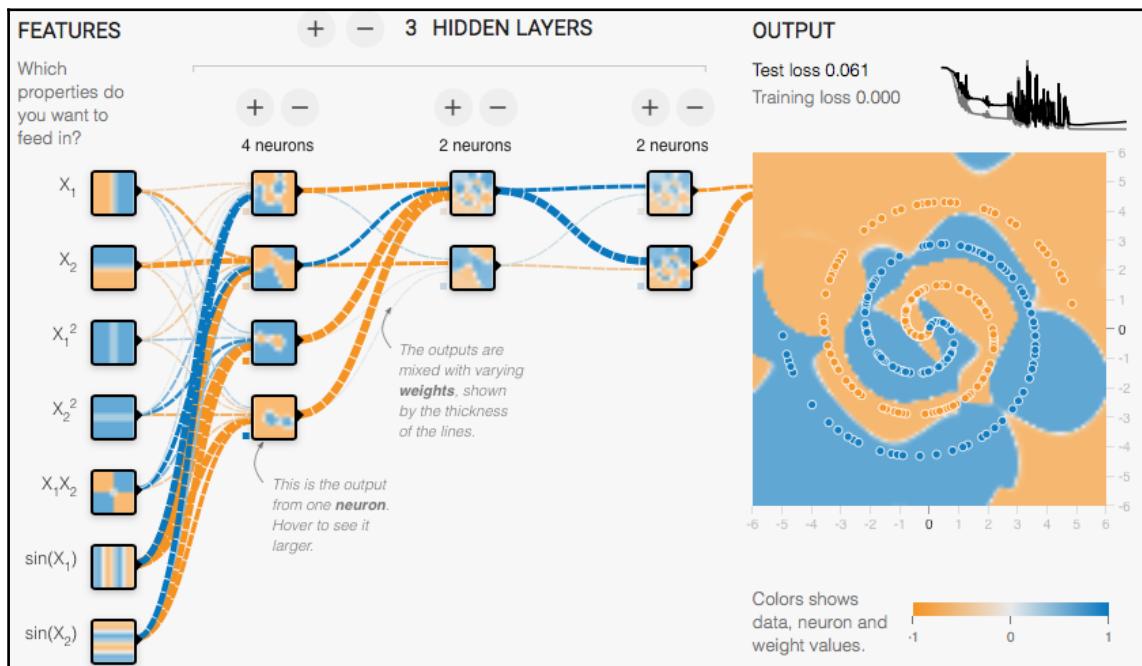
CNN examples



Four layer CNN. Here, an image is convolved to extract large features based on primitives, then using a max pool to scale the image down and feed as input to the feature filters. The fully connected layer ends the CNN path and outputs the best guess.

An example of primitive features that compose the layers are courtesy of TensorFlow (<http://playground.tensorflow.org>). The TensorFlow example of a system with six features on input layer 1, followed by 33 hidden layers of four neurons, followed by two neurons and ending with two more is shown in the following diagram. In this model, the features attempt to classify the color groupings of dots.

Here, we attempt to find the optimal set of features that describe a spiral of two colored balls. The initial feature's primitives are basically lines and stripes. These will combine and be strengthened through the trained weightings to describe the next layer of blobs and splotches. As you move to the right, more detailed and composite representations form. This test ran several thousand epochs in an attempt to show the regions that describe a spiral on the right. You can see the output curve on the upper right, which indicates the amount of error in the training process. Errors actually spiked in the middle of the training run, as chaotic and random effects were seen during backpropagation. The system then healed, and optimized to the final result. The lines between neurons indicate the strength of the weight in describing the spiral pattern:



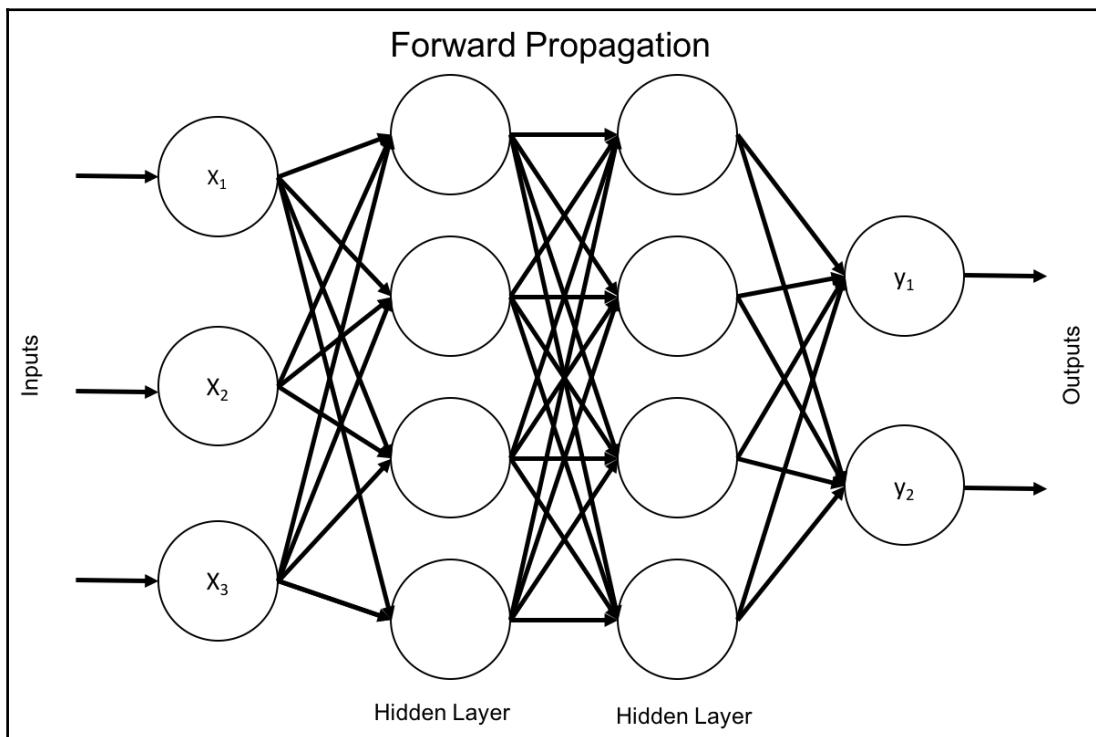
Example CNN in Tensorflow playground. Courtesy of Daniel Smilkov and TensorFlow Playground.

A CNN is modeled in the above image using a learning tool called [Tensorflow Playground](#). Here, the training of a four-layer neural network whose goal is to classify a spiral of different colored balls. The features on the left are the initial primitives, such as horizontal color changes or vertical color changes. The hidden layers are trained through backpropagation. The weighting factor is illustrated by the thickness of a line to the next hidden layer. The result is shown at the right, after several minutes of training.

The last layer is a fully connected layer, so called because it is required that every node in the final layer is connected to every node in the preceding level. The role of the fully connected layer is to finally resolve the image to a label. It does this by inspecting the output and features of the last layer, and determining that the set of the features corresponds to a particular label, such as a car. A car will have wheels, glass windows, and so on; meanwhile a cat will have eyes, legs, furs, and so forth.

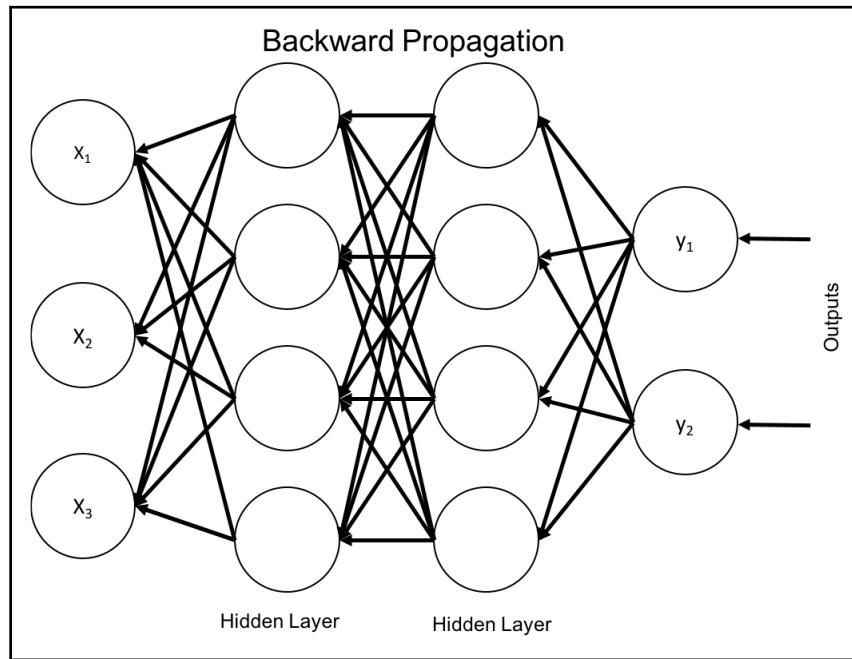
CNN training and backpropagation

We have seen the process of feedforward propagation as a CNN executes. Training a CNN relies on the process of backpropagation of errors and gradients, deriving a new result, and correcting errors over and over. The same network including all pooling layers, activation functions, and matrices are used as the backward propagation flows through the network in attempts to optimize or correct the weighting:



CNN forward propagation during training and inference.

Backpropagation is short for "back propagation of errors". Here an error function will calculate the gradient of an error function based on the neural network weights. The calculation of the gradient is forced backward through all the hidden layers. Shown below is the backpropagation process:



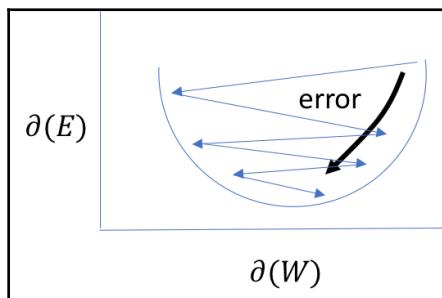
CNN backward propagation during training.

We will now explore the training process. First, we must provide a training set for the network to normalize to. The training set will be talked to later in this chapter but is crucial in developing a well behaving system in the field. The training data will have an image and a known label.

Second, the neural network is composed of identical initial values or random values for each weight on each neuron that needs to be trained. The first forward pass results in substantial errors that go into a loss function:

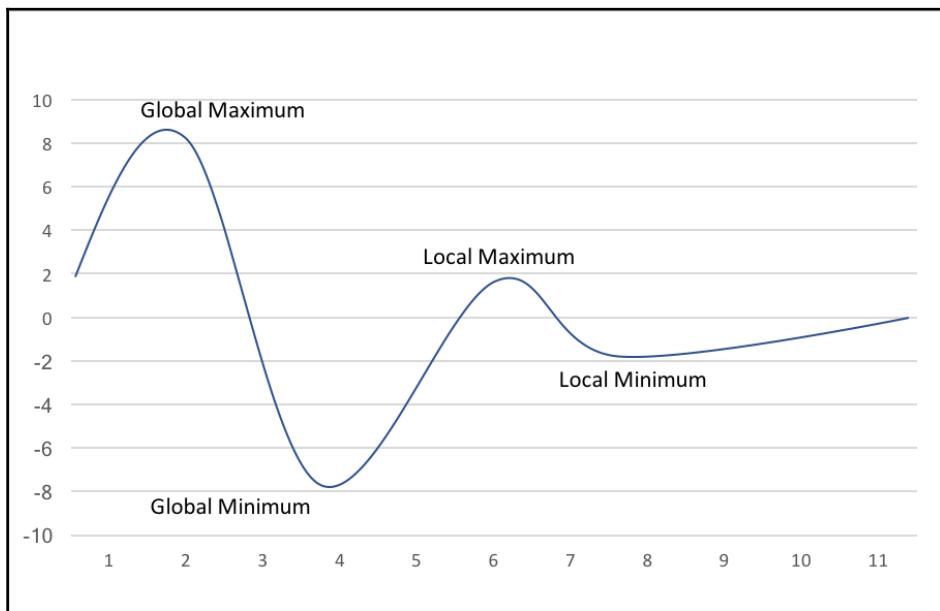
$$W(t) = W(t - 1) - \lambda \times \left(\frac{-\partial E}{\partial W}(t) \right)$$

Here, the new weights are based on the previous weight $W(t - 1)$ minus the partial derivative of the error E over the weight W (loss function). This is also called the **gradient**. In the equation, Lambda refers to the learning rate. This is up to the designer to tune. If the rate is high (greater than 1), the algorithm will use larger steps in the trial process. This may allow the network to converge to an optimal answer faster, or it could produce a poorly trained network that will never converge to a solution. Alternatively, if Lambda is set low (less than 0.01), the training will take very small steps and much longer to converge, but the accuracy of the model may be better. In the following example, the optimal convergence is the very bottom of the curve representing error and weights. If the learning rate is too high, we could never reach the bottom, and will settle for a *near bottom* towards one of the sides:



Global minimum. This illustration shows the basis of a learning function. The goal is to find the minimal value through a gradient descent. The accuracy of the learning model is proportional to the number of steps (time) taken to converge to a minimum.

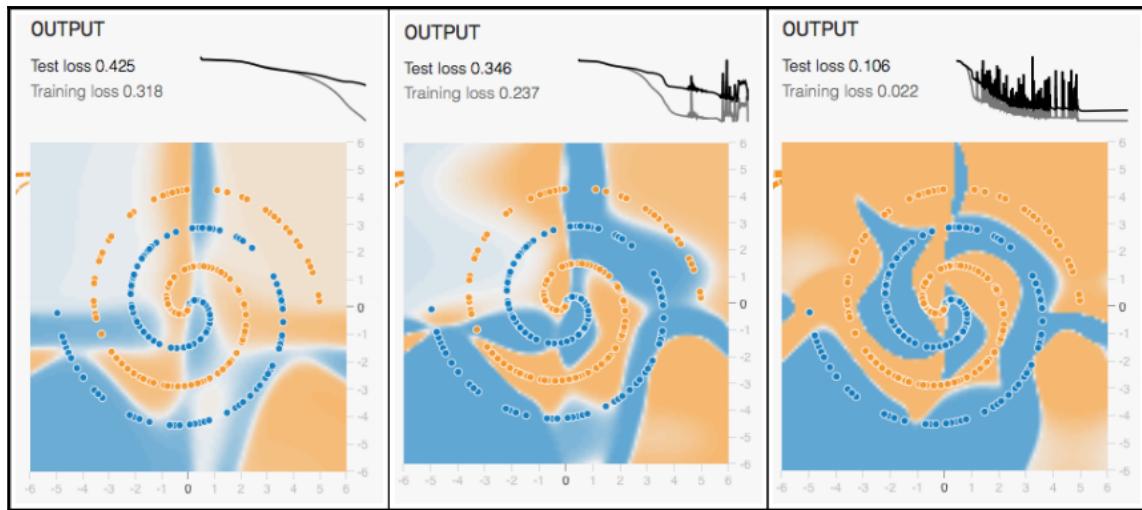
Finding a global minimum of an error function isn't guaranteed. That is, a local minimum may be found and resolved as a false global minimum. The algorithm often will have trouble escaping the local minimum once found. In the next graph, you see the correct global minimum, and how a local minimum may be settled upon:



Errors in training. We see the true global minimum and maximum. Depending on factors such as the training step size or even the initial starting point of the descent, a CNN could be trained to a false minimum.

The loss will be especially heavy during the first initial runs of the network. We can visualize that with the TensorFlow playground. Here again, we are training a neural network to identify spirals. At first in the training, the loss is heavy at **0.425**. After 1531 epochs, we arrive at this network's weights and a loss of **0.106**.

We can see that there is still some degree of error that the training could not resolve:



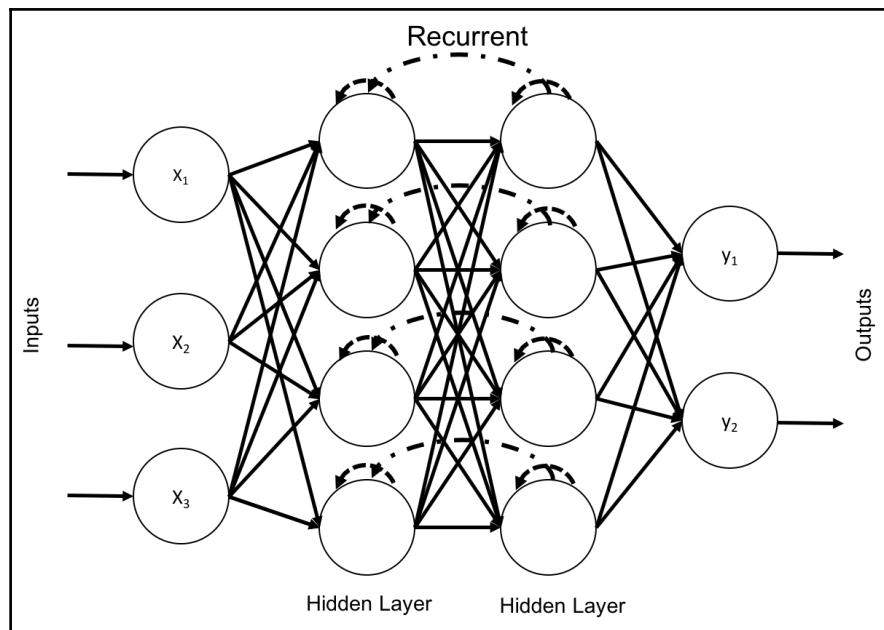
TensorFlow training example. Courtesy of Daniel Smilkov and TensorFlow Playground.

Here, we see the training progress left to right with more accuracy. The left illustration clearly shows the heavy influence of the horizontal and vertical primitive features. After a number of epochs, the training starts converging on the true solution. Even after 1531 epochs, there are still some error cases where the training didn't converge on the correct answer.

RNN

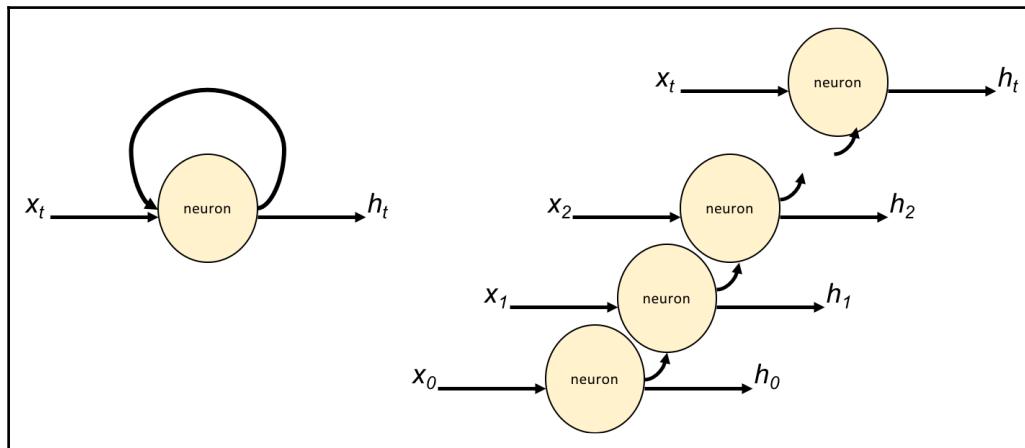
RNNs are a field of machine learning all to themselves and are extremely important and relevant to IoT data. The big difference between an RNN and a CNN is that a CNN processes input on fixed size vectors of data. Think of them as two-dimensional images—that is, a known size input. CNNs also pass from layer to layer as fixed-sized units of data. An RNN has similarities, but is fundamentally different: instead of taking a fixed-size chunk of image data, it has as input a vector and output another vector. At its heart, the output vector is influenced not by that single input we just fed it, but by the entire history of inputs it was fed. That implies an RNN understands the temporal nature of things or can be said to maintain state. There is information to be inferred from the data, but also from the order the data was sent.

RNNs are of particular value in the IoT space especially in time-correlated series of data, such as describing a scene in an image, describing the sentiment of a series of text or values, and classifying video streams. Data may be fed to an RNN from an array of sensors that contain a (time:value) tuple. That would be the input data to send to the RNN. In particular, such RNN models can be used in predictive analytics to find faults in factory automation systems, evaluate sensor data for abnormalities, evaluate timestamped data from electric meters, and even to detect patterns in audio data. Signal data from industrial devices is another great example. An RNN could be used to find patterns in an electrical signal or wave. A CNN would struggle with this use case. An RNN will run ahead and predict what the next value in a sequence will be if the value falls out of the predicted range that could indicate a failure or significant event:



The main difference between an RNN and a CNN is the reference to time or sequence order

If one were to examine a neuron in an RNN, it would look as if it were looping back on itself. Essentially, the RNN is a collection of states going back in time. This is clear if you think of unrolling the RNN at each neuron:



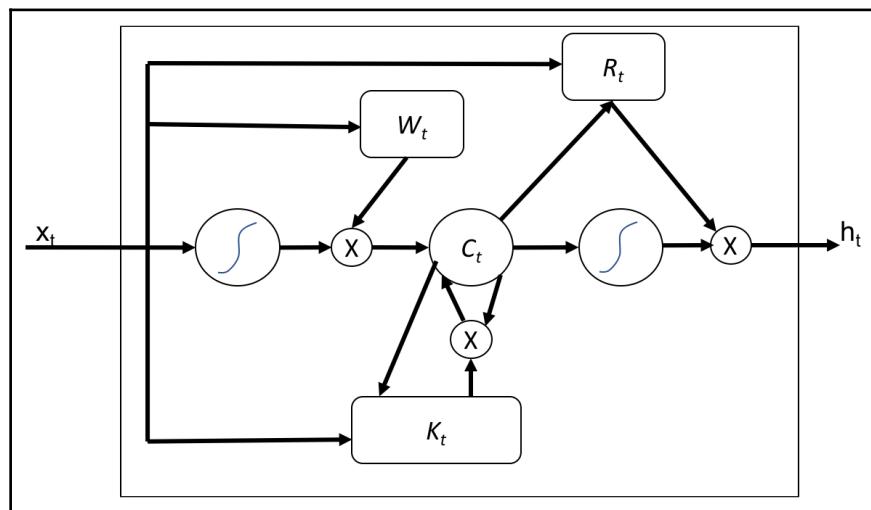
RNN neuron. This illustrates the input from previous step x_{n-1} feeding the next step x_n as the basis of the RNN algorithm.

The challenge with RNN systems is that they are somewhat difficult to train over a CNN or other models. Remember, CNN systems use backpropagation to train and reinforce the model. RNN systems don't have the notion of backpropagation. Anytime we send input into the RNN, it carries a unique timestamp. This leads to the vanishing gradient problem discussed earlier, which reduces the learning rate of the network to be useless. A CNN is also exposed to a vanishing gradient, but the difference with an RNN is that the depth of the RNN can go back many iterations, whereas a CNN traditionally has only a few hidden layers. For example, an RNN resolving a sentence structure like: *A quick brown fox jumped over the lazy dog* will extend back nine levels. The vanishing gradient problem can be thought of intuitively: if weights in the network are small, then the gradient will shrink exponentially leading the vanishing gradient. If the components of the weights are large, then the gradient will grow exponentially and possibly explode (NaN). Exploding leads to an obvious crash, but the gradient is usually truncated or capped before that occurs. A vanishing gradient is harder for a computer to deal with.

One method to overcome this effect is to use the ReLU activation function mentioned in the CNN section. This activation function delivers results 0 or 1, so it isn't prone to vanishing gradients. Another option is the concept of **Long Short-Term Memory (LSTM)**, which was proposed by researchers Sepp Hochreiter and Juergen Schmidhuber. Neural Computation, 9(8):1735-1780, 1997. LSTM solved the vanishing gradient issue, and allowed an RNN to be trained. Here, the RNN neuron consisted of three or four *gates*. These gates allowed the neurons to hold state information, and are controlled by logistic functions with a value between 0 and 1:

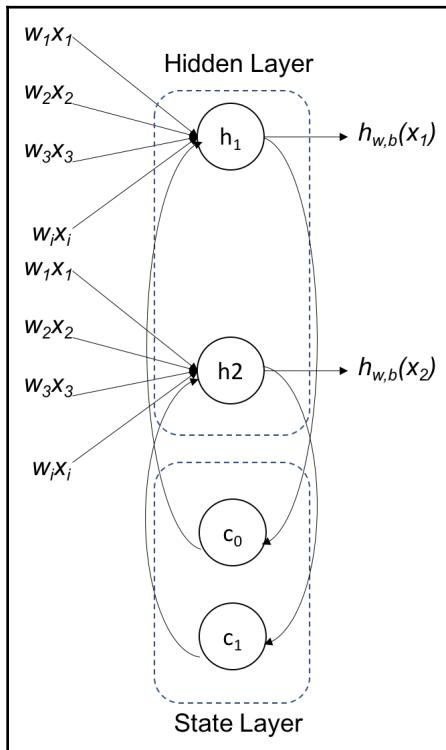
- **Keep gate K:** Controls how much a value will remain in memory
- **Write gate W:** Controls how much a new value will affect memory
- **Read gate R:** Controls how much a value in memory is used to create the output activation function

One can see that these gates are somewhat analog in nature by controlling how much versus a discrete value. The LSTM cell will trap errors in the memory of the cell. This is called the **error carousel** and allows the LSTM cell to backpropagate errors over long time periods. The LSTM cell resembles the following logical structure where the neuron is essentially the same for all outward appearances as a CNN, but internally it maintains state and memory. The LSTM cell of the RNN is illustrated as follows:



LSTM cell. Here is the RNN basic algorithm using an internal memory to process arbitrary sequences of inputs.

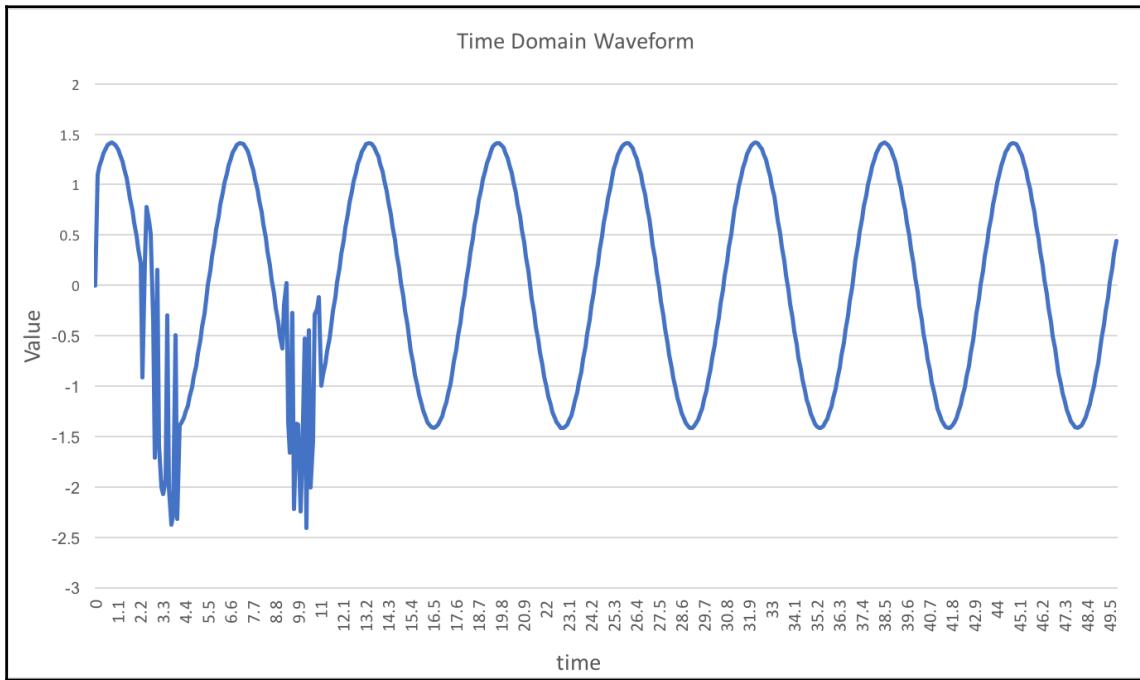
An RNN builds up memory in the training process. This is seen in the diagram as the state layer under a hidden layer. An RNN is not searching for the same patterns across an image or bitmap like a CNN; rather, it is searching for a pattern across multiple sequential steps (which could be time). The hidden layer and state layer complement are shown below:



Hidden layers are fed from previous steps as additional inputs to the next step.

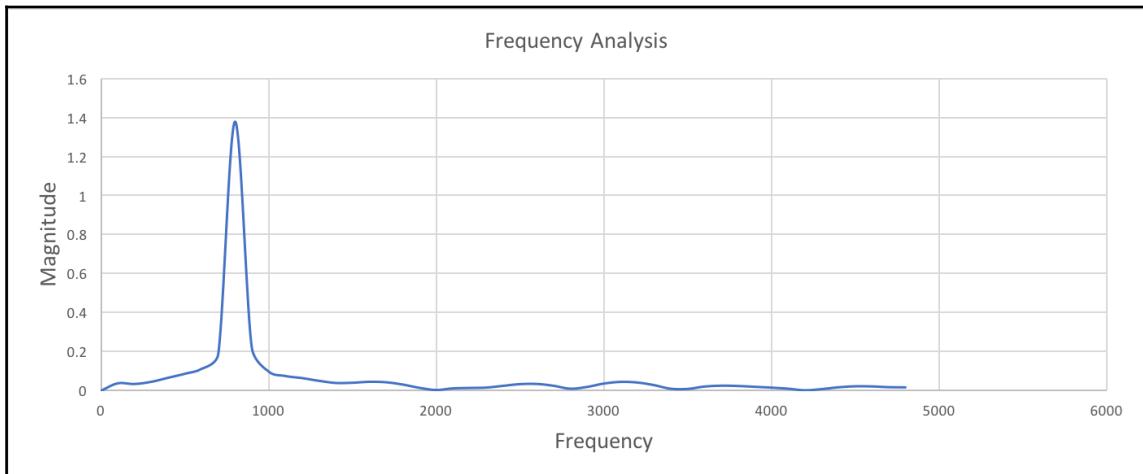
One can see the amount of computation in training with the LSTM logistical math, as well as how the regular backpropagation is heavier than a CNN. The process of training involves backpropagating gradients through the network all the way to time zero. However, the contribution of a gradient from far in the past (say, time zero) approaches zero, and will not contribute to the learning.

A good use case to illustrate an RNN is a signal analysis problem. In an industrial setting, one may historically collect signal data and attempt to infer from it if a machine was faulty, or if there were runaway thermals in some component. A sensor device would be attached to a sampling tool, and a Fourier analysis performed on the data. The frequency components could then be inspected to see if a particular aberration were present. In the following graph, we have a simple sine wave that indicates normal behavior, perhaps of a machine using cast rollers and bearings. We also see two aberrations introduced (the anomaly). A **fast Fourier transform (FFT)** is typically used to find aberrations in a signal based on the harmonics. Here, the defect is a high-frequency spike similar to a Dirac delta or impulse function.



RNN use case. Here, a waveform with an aberration from audio analysis could be used as input to an RNN.

We see the following FFT only registers the carrier frequency and doesn't see the aberration:



Hogh frequency spike via an FFT

An RNN specifically trained to identify the time-series correlation of a particular tone or audio sequence is a straightforward application. In this case, an RNN could replace an FFT, especially when multiple sequences of frequencies or states are used to classify a system, making it ideal for sound or speech recognition.

Industrial predictive maintenance tools rely on this type of signal analysis to find thermal and vibration-based failures of different machines. This traditional approach has limits, as we see. A machine learning model (especially an RNN) can be used to inspect the incoming stream of data for particular feature (frequency) components, and could find point failures as shown in the preceding graph. Raw data, shown in the previous graph, is arguably never as clean as a sine wave. Usually, the data is quite noisy with periods of loss.

Another use case is around sensor fusion in healthcare. Healthcare products like glucose monitors, heart rate monitors, fall indicators, respiratory meters, and infusion pumps will send period or a stream of data. All these sensors are independent of each other, but together comprise a picture of patient health. They also are time-correlated. An RNN can bridge this unstructured data in aggregate, and predict patient health all dependent on the patient's activity throughout the day. This can be useful for home health monitoring, sports training, rehabilitation, and geriatric care.

One must be careful with RNNs. While they can make good inference on time series data, and they can predict oscillations and wave behaviors, they may behave chaotically, and they are very difficult to train.

Training and inference for IoT

While neural networks offer significant advantages of having a machine behave closer to a human in the areas of perception, pattern recognition, and classification, it is primarily up to the training to develop a model that works well with low loss, no overfitting, and adequate performance. In the IoT world, latency is a large issue, especially for safety-critical infrastructure. Resource constraints is another factor. Most of the edge computing devices that exist today do not have hardware accelerators such as **General-Purpose Computation on Graphics Hardware (GPGPU)** and **Field-programmable gate array (FPGAs)** at their disposal to assist with the heavy matrix math and floating point surrounding neural networks. Data could be sent to the cloud, but that may have significant latency effects as well as bandwidth costs. The OpenFog group is provisioning a framework where edge fog nodes may be provisioned with additional compute resources, and pulled on demand to provide assistance with the heavy lifting of these algorithms.

Training, for now, should be the realm of the cloud, where the computing resources are available and test sets can be created. The edge devices should report to the cloud parent when a training model fails, or if new data appears that requires a retraining effort. The cloud allows for a *train once-deploy many* concept, which is a strength. Alternatively, it is wise to consider training on a regional basis with bias. The concept here is that fog nodes in particular regions may be more sensitive to certain patterns that are environmentally different. For example, monitoring temperature and humidity on equipment in the field in the Arctic will differ significantly to a tropical region.

The edge is more adept at running trained models in an inference mode. However, deploying inference engines needs to be architected well. Some CNN networks such as AlexNet have 61 million parameters, consume 249 MB of memory, and perform 1.5 billion floating point operations to classify a single image.

Reduced precision, pruning, and other techniques to perform first-run heuristics on image data are better suited for edge devices. Additionally, preparing data for upstream analytics can also help. Examples include:

- **Sending upstream:** Only data that meets a specific condition (time, event of interest)
- **Data scrubbing:** Reduce, crop, clip data set to only relevant content
- **Segment:** Force data into grayscale to reduce traffic and prepare it for CNN

IoT data analytics and machine learning comparison and assessment

The uses of a machine learning algorithm have their place in IoT. The typical case is when there is a plethora of streaming data that needs to produce some meaningful conclusion. A small collection of sensors may only need a simple rule engine on the edge in a latency-sensitive application. Others may stream data to a cloud service and apply rules there for systems with less-aggressive latency demands. When large amounts of data, unstructured data, and real-time analytics come into play, we need to consider the use of machine learning to solve some of the hardest problems.

In this section, we detail some tips and reminders in deploying machine learning analytics, and what use cases may warrant such tools.

Training phase:

- For random forest, use bagging techniques to create ensembles.
- When using a random forest, ensure you maximize the number of decision trees.
- Watch overfitting. Overfitting will lead to inaccurate field models. Techniques such as regularization and even injecting noise into a system will reinforce the mode.
- Don't train on the edge.
- Gradient descent will lead to error. RNNs naturally are susceptible.

Model in field:

- Update model with new data sets as they become available. Keep the training set current.
- Running models on the edge can be reinforced with larger and more comprehensive models in the clouds.
- Neural network execution can be optimized in the cloud and at the edge with a minimum loss by considering techniques such as pruning node and reducing precision.

Model	Best application	Worse fit and side effects	Resource demands	Training
Random forests (statistical models)	<ul style="list-style-type: none"> • Anomaly detection • Systems with 1000's of choice points and hundreds of inputs • Regression and classification • Handles mixed data types • Ignores missing values • Scales linearly with input 	<ul style="list-style-type: none"> • Feature extraction • Time and sequence analysis 	Low	<ul style="list-style-type: none"> • Training based on bagging techniques, for maximum effectiveness • Training fairly resource light • Mainly supervised
RNN (temporal and sequence-based neural networks)	<ul style="list-style-type: none"> • Prediction of an event based on a sequence • Streaming data patterns • Time-correlated series data • Maintains knowledge of past states to predict new states (electrical signals, audio, speech recognition) • Unstructured data • Input variables may or may not be dependent 	<ul style="list-style-type: none"> • Image and video analysis • Systems of requiring thousands of features 	<ul style="list-style-type: none"> • Very high for training • High for inference execution 	<ul style="list-style-type: none"> • Training more cumbersome than CNN backpropagation • Very hard to train • Supervised
CNN (deep learning)	<ul style="list-style-type: none"> • Prediction of an object based on surrounding values • Pattern and feature identification • 2D image recognition • Unstructured Data • Input variables may or may not be dependent 	<ul style="list-style-type: none"> • Time-based and sequential predictions • Systems of requiring thousands of features 	<ul style="list-style-type: none"> • Very high for Training (floating point precision, large training sets, large memory demands) • High for inference execution 	Supervised and unsupervised
Bayesian networks (probabilistic models)	<ul style="list-style-type: none"> • Noisy and incomplete data sets • Streaming data patterns • Time correlated series • Structured data • Signal analysis • Models developed quickly 	<ul style="list-style-type: none"> • Assumes all input variables are independent • Perform poorly with high orders of data dimensions 	Low	<ul style="list-style-type: none"> • Little training data need with respect to other artificial neural networks

Summary

This chapter was a brief introduction to data analytics for IoT in the cloud and in the fog. Data analytics is where the value is extracted out of the sea of data produced by millions or billions of sensors. Analytics is the realm of the data scientist in attempts to find the hidden patterns and develop predictions from an overwhelming amount of data. To be valuable, all this analysis needs to be at or near real time to make life-critical decisions. One needs to understand the problem being solved, and the data necessary to reveal the solution. Only then can a data analysis pipeline be architected well. This chapter exposed several data analysis models as well as an introduction to the four relevant machine learning domains. These analytics tools are the heart of value in IoT to derive meaning from the nuances of massive amounts of data in real time. Machine learning models can predict future events based on current and historical patterns. We see how RNN and CNN cases satisfy this context through proper training. As an architect, the pipeline, storage, models, and training all need to be considered.

In the next chapter, we will talk about security of IoT from a holistic point of view, from the sensor to the cloud. We will examine specific real-world attacks on the IoT in recent years, as well as methods to counter such attacks in the future.

12

IoT Security

The first chapter of this book revealed the size, growth, and potential of the **Internet of Things (IoT)**. There are currently billions of devices, and the double-digit growth of connecting the analog world to the internet also forms the largest attack surface on Earth. Exploits, damage, and rogue agents have already been developed, deployed, and spread globally, disrupting countless businesses, networks, and lives. As architects, we are responsible for understanding the IoT stack of technologies and securing them. As we place devices that have never been connected to the internet, as good citizens, we are accountable for designing them.

This has been particularly difficult for many IoT deployments with security often being thought of last. Often, systems are so constrained that building enterprise-level security that modern web and PC systems enjoy is difficult if not impossible on simple IoT sensors. This book also talks about security after all other technologies have been understood. However, every chapter has touched on the provisions of security at each level.

This chapter will explore some particularly heinous IoT focused attacks and give thought to how weak security is in IoT and how much damage can be done. Later, we discuss the security provisions at each level of the stack: physical devices, communication systems, and networks. We then address software-defined perimeters and blockchains used to secure value in IoT data. The chapter wraps up by examining the United States Cybersecurity Improvement Act of 2017 and what it could mean for IoT devices.

The most important thing in security is to use it at all levels from the sensor to the communication system, router, and cloud.

Cyber security vernacular

Cybersecurity has an associated set of definitions describing different types of attacks and provisions. This section briefly covers the jargon of the industry as presented in the rest of this chapter.

Attack and threat terms

The following are the terms and definitions of different attacks or malevolent cyber threats:

- **Amplification attack:** Magnifies the bandwidth sent to a victim. Often an attacker will use a legitimate service such as NTP, Steam, or DNS to reflect the attack upon a victim. NTP can amplify 556x and DNS amplification can escalate the bandwidth by 179x.
- **ARP spoof:** A type of attack that sends a falsified ARP message resulting in linking the attacker's MAC address with the IP of a legitimate system.
- **Banner scans:** A technique typically used to take inventory of systems on a network that can also be used by an attacker to gain information about a potential attack target by performing HTTP requests and inspecting the returned information of the OS and computer (for example, nc www.target.com 80).
- **Botnets:** Internet-connected devices infected and compromised by malware working collectively by common control, mostly used in unison to generate massive DDoS attacks from multiple clients. Other attacks include email spamming and spyware.
- **Brute force:** A trial and error method to gain access to a system or bypass encryption.
- **Buffer overflow:** Exploits a bug or defect in running software that simply overruns a buffer or memory block with more data than allocated. This overrun can write over other data in adjacent memory addresses. An attacker can lay malicious code in that area and force the instruction pointer to execute from there. Compiled languages such as C and C++ are particularly susceptible to buffer overflow attacks since they lack internal protection. Most overflow bugs are the result of poorly constructed software that does not check the bounds of input values.
- **C2:** Command and control server that marshals commands to botnets.

- **Correlation power analysis attack:** Allows one to discover secret encryption keys stored in a device through four steps. First, examine a target's dynamic power consumption and record it for each phase of the normal encryption process. Next, force the target to encrypt several plaintext objects and record their power usage. Next, attack small parts of the key (subkeys) by considering every possible combination and calculating the Pearson correlation coefficient between the modeled and actual power. Finally, put together the best subkey to obtain the full key.
- **Dictionary attack:** A method of gaining entry to a network system by systematically entering words from a dictionary file containing the username and password pairs.
- **Distributed Denial of Service (DDoS):** An attack attempting to disrupt or make an online service unavailable by overwhelming it from multiple (distributed) sources.
- **Fuzzing:** A fuzzing attack consists of sending malformed or non-standard data to a device and observing how the device reacts. For example, if a device performs poorly or shows adverse effects, the fuzz attack may have exposed a weakness.
- **Man-in-the-Middle Attack (MITM):** A common form of attack that places a device in the middle of a communication stream between two unsuspecting parties. The device listens, filters, and appropriates information from the transmitter and retransmits selected information to the receiver. A MITM may be in the loop acting as a repeater or can be sideband listening to the transmission without intercepting the data.
- **NOP sleds:** A sequence of injected NOP assembly instructions used to "slide" a CPU's instruction pointer to the desired area of malicious code. Usually part of a buffer overflow attack.
- **Replay attack (also known as a playback attack):** A network attack where data is maliciously repeated or replayed by the originator or an adversary who intercepts the data, stores the data, and transmits it at will.
- **RCE exploit:** Remote code execution that enables an attacker to execute arbitrary code. This usually comes in the form of a buffer overflow attack over HTTP or other network protocols that injects malware code.
- **Return-Oriented Programming (ROP Attack):** This is a difficult security exploit an attacker may use to potentially subvert protections with non-executing memory or executing code from read-only memory. If an attacker gains control of a process stack through a buffer overflow or some other means, they may jump to legitimate and unchanged sequences of instructions already present. The attacker looks for sequences of instructions to call gadgets that can be pieced together to form a malevolent attack.

- **Return-to-libc:** A type of attack that starts with a buffer overflow where the attacker injects jumps to libc or other popularly used libraries in the processes' memory space in an attempt to call system routines directly. Bypasses the protection offered by non-executable memory and guard bands. This is a specific form of ROP attack.
- **Rootkit:** Typically malicious software (although often used to unlock smartphones) used to enable other software payloads to be undetectable. Rootkits use several targeted techniques such as buffer overflows to attack kernel services, hypervisors, and user mode programs.
- **Side Channel Attack:** An attack used to gain information from a victim's system by observing the secondary effects of the physical system rather than find runtime exploits or zero-day exploits. Examples of side channel attacks include correlation power analysis, acoustic analysis, and reading data residue after it has been deleted from memory.
- **Spoofing:** Malicious party or device impersonates another device or user on a network.
- **SYN flood:** Occurs when a host sends a TCP:SYN packet which a rogue agent will spoof and forge. This will cause the host to create half-open connections to many non-existent addresses causing the host to exhaust all resources.
- **Zero-Day exploits:** Security defects or bugs in commercial or production software unknown to the designer or manufacturer.

Defense terms

The following are the terms and definitions of different cyber defense mechanisms and technologies:

- **Address Space Layout Randomization:** Also known as **ASLR**, this defense mechanism protects memory and thwarts buffer overflow attacks by randomizing where an executable is loaded in memory. A buffer overflow injecting malware can not predict where it will be loaded in memory, thus manipulating the instruction pointer will become extremely challenging. Protects against return-to-libc attacks.
- **Black hole (sinkhole):** After detecting a DDoS attack, routes are established from the affected DNS server or IP address to force rogue data to a black hole or a non-existent endpoint. Sinkholes perform further analysis to filter out good data.

- **Data Execution Prevention (DEP):** Marks an area as executable or non-executable. This prevents an attacker from running code maliciously injected into such a region via a buffer overflow attack. The result is a system error or exception.
- **Deep Packet Inspection (DPI):** A method of inspecting each packet (data and possibly header information) in a data stream to isolate intrusions, viruses, spam, and other criteria being filtered.
- **Firewall:** A network security construct that grants or rejects network access to packet streams between an untrusted zone and a trusted zone. Traffic can be controlled and managed through **access control lists (ACL)** on routers. Firewalls can perform stateful filtering and provide rules based on destination ports and traffic state.
- **Guard bands and non-executable memory:** Protects regions of memory that are writeable and not executable. Protects against NOP sleds. Intel: NX bit, ARM XN bit.
- **Honeypots:** Security tool to detect, deflect, or reverse engineer malicious attacks. Honeypots appear as legitimate websites or accessible nodes in a network but are actually isolated and monitored. Data and interactions with the device are logged.
- **Instruction-Based Memory Access Control:** A technique to separate the data portion of a stack from the return address portion. This technique helps protect against ROP attacks and is particularly useful in constrained IoT systems.
- **Intrusion Detection System (IDS):** A network construct to detect threats in a network through the out-of-band analysis of the packet stream therefore not in-line with the source and destination so as to affect real-time response.
- **Intrusion Prevention System:** Blocks threats to a network via true in-line analysis and statistical or signature detection of threats.
- **Milkers:** A defensive tool that emulates an infected botnet device and attaches to its malevolent host allowing one to understand and "milk" the malware commands being sent to the controlled botnet.
- **Port scanning:** A method to find an open and accessible port on a local network.
- **Public Key Infrastructure (PKI):** Provides a definition of hierarchies of verifiers to guarantee the origin of a public key. A certificate is signed by certificate authorities.
- **Public key:** A public key is generated with a private key and is accessible to external entities. A public key can be used to decrypt hashes.

- **Private key:** A private key is generated with a public key, never released externally, and stored securely. It is used to encrypt hashes.
- **Root of Trust (RoT):** Starts execution on a cold boot device from an immutable trusted source of memory (such as ROM). If early boot software/BIOS can be changed without control, then no Root of Trust exists. The Root of Trust is usually the first phase in a multi-phase secure boot.
- **Secure Boot:** A series of boot steps for a device that starts at a Root of Trust and proceeds through OS and application loading where each component signature is verified as authentic. Verification is performed through public keys loaded at previous trusted boot stages.
- **Stack canaries:** Guards process stack space from stack overruns and prevents the execution of code from a stack.
- **Trusted Execution Environment:** A secure area of a processor that ensures code and data residing within this zone is protected. This is usually an execution environment on the main processor core where the code for secure booting, monetary transfers, or private key handling will be executed with a higher level of security than the majority of the code.

Anatomy of IoT cyber attacks

The area of cybersecurity is a broad and massive topic beyond the scope of this chapter. However, it is useful to understand three types of IoT-based attacks and exploits. Since the topology of the IoT consists of hardware, networking, protocols, signals, cloud components, frameworks, operating systems, and everything in-between, we will now detail three forms of prevalent attacks:

- **Mirai:** The most damaging denial of service attack in history that spawned from insecure IoT devices in remote areas.
- **Stuxnet:** A nation-state cyber weapon targeting industrial SCADA IoT devices controlling substantial and irreversible damage to Iran's nuclear program.
- **Chain Reaction:** A research method to exploit PAN area networks using nothing but a lightbulb—no internet needed.

By understanding the behaviors of these threats, the architect can derive preventative technologies and processes to ensure similar events are mitigated.

Mirai

Mirai is the name of malware that infected Linux IoT devices in August of 2016. The attack came in the form of a botnet that generated a massive DDOS storm. High-profile targets included Krebs on Security, a popular internet security blog, Dyn, a very popular and widely used DNS provider for the internet, and Lonestar cell, a large telecom operator in Liberia. Smaller targets included Italian political sites, *Minecraft* servers in Brazil, and Russian auction sites. The DDOS on Dyn had secondary effects on other extremely large providers that used their services such as Sony Playstation servers, Amazon, GitHub, Netflix, PayPal, Reddit, and Twitter. In total, 600,000 IoT devices were infected as part of the botnet collective.

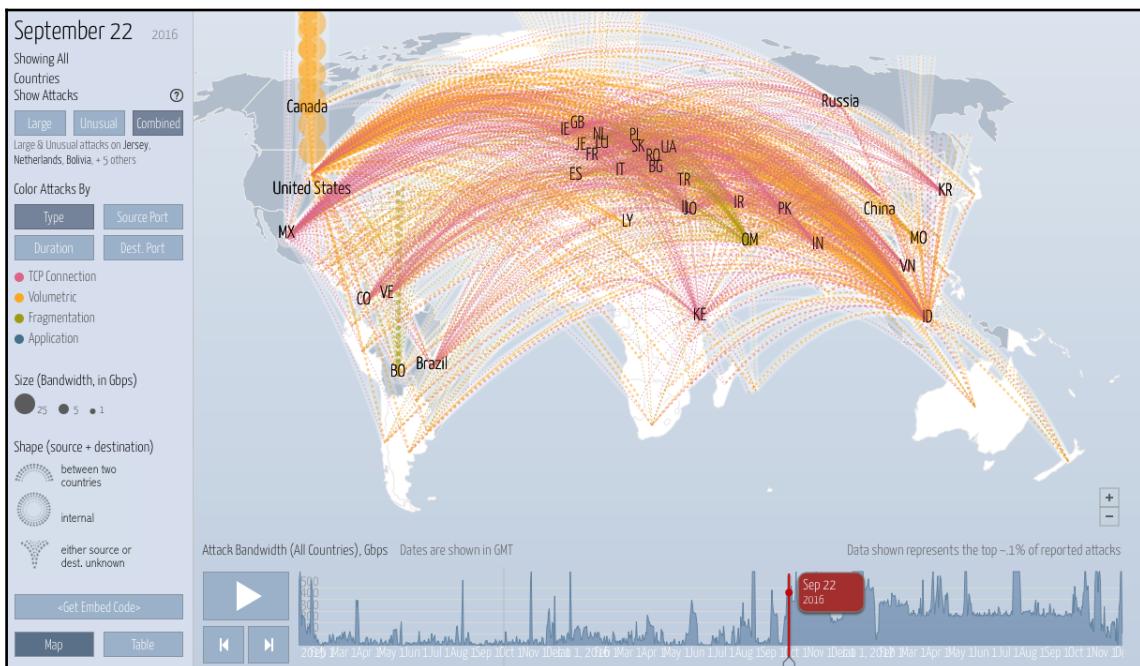
Mirai source code was released on hackforums.net (a hacker blog site). From the source and through traces and logs, researchers have uncovered how the Mirai attack worked and unfolded:

1. **Scan for victims:** Perform a rapid asynchronous scan using TCP SYN packets to probe random IPV4 addresses. It specifically looked for SSH/Telnet TCP port 23 and port 2323. If the scan and port attached successfully, it entered phase two. Mirai included a hardcoded blacklist of addresses to avoid. The blacklist consisted of 3.4 million IP addresses and contained IP addresses belonging to the US Postal Services, Hewlett Packard, Ge, and the Department of Defense. Mirai was capable of scanning at about 250 bytes per second. This is relatively low as far as a botnet is concerned. The attacks like the SQL Slammer generated scans at 1.5 Mbps, the principle reason being that IoT devices typically are much more constrained in processing power than desktop and mobile devices.
2. **Brute Force Telnet:** At this point Mirai attempted to establish a functional Telnet session with a victim by sending 10 username and password pairs randomly using a dictionary attack of 62 pairs. If a login was successful, Mirai logged the host to a central C2 server. Later variants of Mirai evolved the bot to perform RCE exploits.
3. **Infect:** A loader program was then sent to the potential victim from the server. It was responsible for identifying the operating system and installing device-specific malware. It then searched for other competing processes using port 22 or 23 and killed them (along with other malware that could already be present on the device). The loader binary was then deleted and the process name was obfuscated to hide its presence. The malware did not reside in persistent storage and didn't survive a reboot. The bot now stayed dormant until it received an attack command.

The devices targeted were IoT devices comprising IP cameras, DVRs, consumer routers, VOIP phones, printers, and set-top boxes. These consisted of 32-bit ARM, 32-bit MIPS, and 32-bit X86 malware binaries specific to the IoT device being hacked.

The first scan occurred on August 1, 2016, from a US web hosting site. The scan took 120 minutes before it found a host with an open port and password in the dictionary. After one additional minute, 834 other devices were infected. Within 20 hours, 64,500 devices were infected. Mirai doubles in size in 75 minutes. Most of the infected devices that turned into botnets were located in Brazil (15.0%), Columbia (14.0%), and Vietnam (12.5%), although the targets of the DDOS attacks were in other regions.

The damage was confined to DDOS attacks. The DDOS attacks came in the form of SYN floods, GRE IP network floods, STOMP floods, and DNS floods. Over the course of five months, 15,194 individual attack commands were issued by the C2 servers and hit 5,042 internet sites. On September 21, 2016, the Mirai botnet unleashed a massive DDOS attack on the Krebs on Security blog site and generated 623 Gbps of traffic. That accounted for the single worst DDOS attack of all time. The following is a real-time screenshot captured during the Mirai attack using www.digitalattackmap.com: a collaboration between NETSCOUT Arbor and Google Jigsaw.



A view of the Mirai DDoS attack on the Krebs on Security website; courtesy of www.digitalattackmap.com

Stuxnet

Stuxnet was the first known documented cyber weapon released to permanently damage another nation's assets. In this case, it was a worm that was released to damage SCADA-based Siemens **Programmable Logic Controllers (PLC)** and used a rootkit to modify the rotational speed of motors under the direct control of the PLC. The designers went out of their way to ensure the virus targeted only devices with rotational spin rates of slave variable frequency drives attached to Siemens S7-300 PLCs rotating at 807 Hz and 1210 Hz, as they are typically used for pumps and gas centrifuges for uranium enrichment.

The attack presumably started in April or March of 2010. The infection process followed these steps:

1. **Initial infection:** The worm started by infecting a host Windows machine exploiting vulnerabilities found in previous virus attacks. It is thought to have spread by the insertion of a USB drive in the initial machine. It used four zero-day exploits simultaneously (this is an unprecedented level of sophistication). The exploits used a rootkit attack using user-mode and kernel-mode code and installed a stolen yet properly signed and certified device driver from Realtek. This kernel-mode signed driver was necessary to hide Stuxnet from wary antivirus packages.
2. **Windows attack and spread:** Once installed through the rootkit, the worm began to search the Windows system for files typical of a Siemens SCADA controller, WinCC/PCS 7 SCADA, also known as Step-7. If the worm happened to find Siemens SCADA control software, it attempted to access the internet through a C2 using malformed URLs (www.mypremierfutbol.com and www.todaysfutbol.com) to download more recent versions of its payload. It then dug further into the filesystem to search for a file called `s7otbdx.dll`, which served as a critical communications library between the Windows machine and the PLC. Step-7 included a hardcoded password database which was exploited through another zero-day attack. Stuxnet inserted itself between the WinCC system and the `s7otbdx.dll` to act as a Man-in-the-Middle attacker. The virus started its operation by recording the normal operation of the centrifuges.
3. **Destruction:** When it decided to coordinate the attack, it replayed the pre-recorded data to the SCADA systems which had no reason to believe anything was compromised or behaving erratically. Stuxnet delivered its damage by manipulating the PLCs with two different coordinated attacks to damage the entire array of Iran's facility. The damage to the rotors of the centrifuge occurred slowly over time, running in 15 or 50-minute increments separated by 27 days of normal operation. This resulted in improperly enriched uranium as well as cracked and destroyed rotor tubes in the centrifuges.

It is believed that over 1,000 uranium enrichment centrifuges were crippled and damaged by this attack on Iran's main enrichment facility in Natanz, Iran. Today the Stuxnet code is available online and is essentially an open source playing field to create derivative exploits (<https://github.com/micriktor/stuxnet>).

Chain Reaction

Chain Reaction is an academic study that shows a new breed of cyber attacks focused on PAN mesh networks which can be executed without any link to the internet. Additionally, it shows how vulnerable remote IoT sensor and control systems can be. The attack vector was Philips Hue light bulbs typically found in consumer homes that can be controlled by the internet and smartphone apps. The exploit can be scaled up to smart city attacks and initiated by simply inserting one single infected smart light.

Philips Hue lights use the Zigbee protocol to establish a mesh. Zigbee lighting systems fall under a program called the **Zigbee Light Link (ZLL)** to force a standard method for lighting interoperability. ZLL messages are not encrypted or signed but encryption is used to secure keys exchanged if a light is added to the mesh. This master key is known to everyone in the ZLL alliance and was subsequently leaked. ZLL also forces light bulbs joining the mesh to be in very close proximity to the initiator. This prevents one from taking over their neighbor's lights. Zigbee also offers an **Over-the-Air (OTA)** reprogramming method; however, the firmware bundles are encrypted and signed.

The researchers used a four-phase attack plan:

1. Break the encryption and signing of the OTA firmware bundle.
2. Write and deploy a malevolent firmware upgrade to a single light bulb using the broken encryption and signing keys.
3. The compromised bulb would join the network based on the stolen master key and exploit the proximity security through a found zero-day defect in the popularly used Atmel AtMega part.
4. After successfully joining a Zigbee mesh, it would then send its payload to neighboring lights and infect them rapidly. This would expand based on *Percolation Theory* and infect entire city populations of lighting systems.

Zigbee uses AES-CCM (part of IEEE 802.15.4 standard and covered later in this chapter) encryption to encrypt OTA firmware updates. To break the firmware encryption, the attackers used **Correlation Power Analysis (CPA)** and **Differential Power Analysis (DPA)**. This is a sophisticated form of attack where a device such as the light bulb controller hardware is placed on a bench and the power that it consumes is measured. Given sophisticated control, one can measure the dynamic power used by a CPU executing an instruction or moving data (for example, when an encryption algorithm is executed). This is called simple power analysis, in which it is still be very difficult to crack the key. CPA and DPA extend capabilities beyond simple power analysis by using a statistical correlation. Rather than attempt to determine one bit at a time in cracking a key, CPA can resolve byte-wide quantities. Power traces are captured by an oscilloscope and split into two sets. The first set assumes an intermediate value being cracked is set to 1 and the other set assumes it is set to 0. By subtracting the mean of these sets, the true value of an intermediate value is exposed.

Using both DPA and CPA, the researchers broke the Philips Hue lighting system as follow:

- Used the CPA to crack the AES-CBC. The attackers had no key, no nonce, no initialization vector. This resolved the key, which was used in the same manner to attack the nonce.
- Used DPA to crack the AES-CTR counter mode to break the firmware bundling encryption. Researchers found 10 locations that the AES-CTR seemed to execute which created 10x the possibilities.
- Researchers then focussed on breaking the Zigbee proximity protection for joining a network. The zero-day exploit was the result found by inspecting Atmel's source code for the bootloader on the SOC. Upon reviewing the code, they found that the proximity check was valid when starting a scan request in Zigbee. If they started with any other message, the proximity check was bypassed. This allowed them to join any network.

A true attack could force an infected bulb to infect others within a few hundred meters with a payload to remove the firmware update ability of each bulb so they can never be recovered. The bulbs would effectively be under malicious control and would have to be destroyed. The researchers were able to build a fully automated attack system and attached it to a drone that systematically flew within range of Philips Hue lights in a campus environment and hijacked each one.



More information on the CPA attack on Zigbee can be found here: *E. Ronen, A. Shamir, A. O. Weingarten and C. O'Flynn, "IoT Goes Nuclear: Creating a ZigBee Chain Reaction," 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, 2017, pp. 195-212.* An excellent tutorial and the source code to generate a CPA attack can be found on the ChipWhisperer Wiki: https://wiki.newae.com/AES-CCM_Attack.

Physical and hardware security

Many IoT deployments will be in remote and isolated areas leaving sensors and edge routers vulnerable to physical attack. Additionally, the hardware itself needs modern protection mechanisms common in processors and the circuitry of mobile devices and personal electronics.

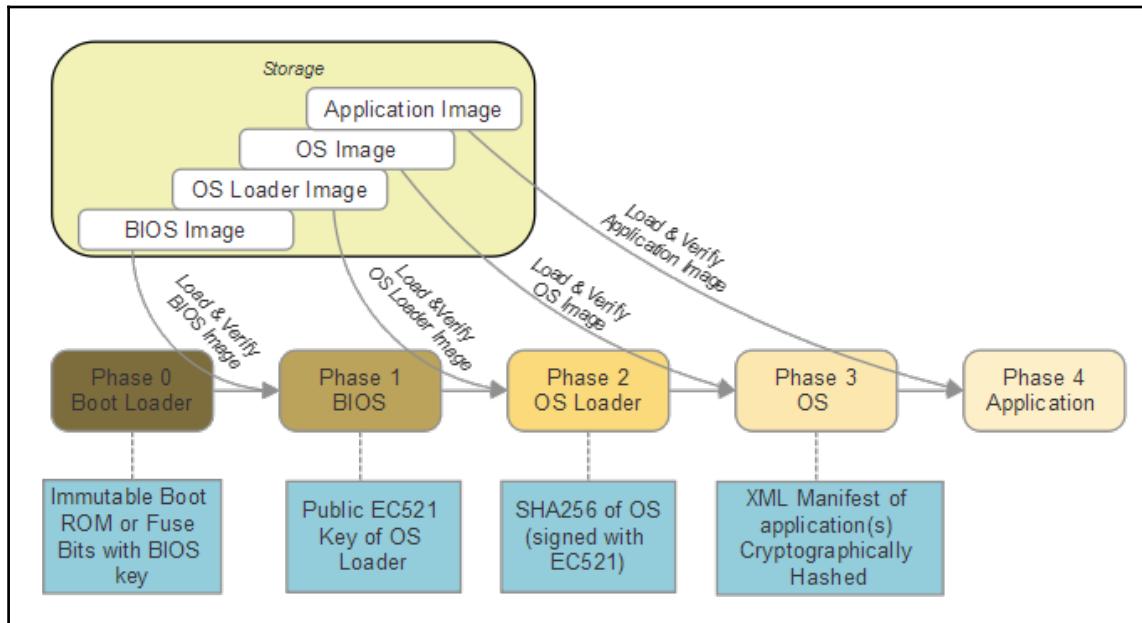
Root of Trust

The first layer of hardware security is the establishment of a Root of Trust. The **Root of Trust (RoT)** is a hardware-validated boot process that ensures the first executable opcode starts from an immutable source. This is the anchor of the boot process that subsequently plays a role in bootstrapping the rest of the system from BIOS to the operating system to the application. A RoT is a baseline defense against a rootkit.

Each phase validates the next phase in the boot process and builds a Chain of Trust. An RoT can have different starting methods such as:

- Boot from ROM or a non-writable memory to store the image and root key
- One-time programmable memory using fuse bits for root key storage
- Boot from a protected memory region that loads code into a protected memory store

An RoT also needs to validate each new phase of boot. Each phase of boot maintains a set of cryptographically signed keys that are used to verify the next phase of boot:



Establishing a Root of Trust. Here is a five-phase boot building up a Chain of Trust and starting with a Boot Loader in immutable read-only memory. Each phase maintains a public key that is used to verify the authenticity of the next component loaded.

Processors that support an RoT are architecturally unique. Intel and ARM support the following:

- **ARM TrustZone**: ARM sells a security silicon IP block for SOC manufacturers that provides a hardware Root of Trust as well as other security services. TrustZone divides hardware into secure and non-secure "worlds". TrustZone is a separated microprocessor from the non-secure core. IT runs a Trusted OS specially designed for security that has a well-defined interface to the non-secure world. Protected assets and functions reside in the trusted core and should be lightweight by design. The switching between worlds is done through hardware context switching, eliminating the need for secure monitor software. Other uses for TrustZone are to manage system keys, credit card transactions, and Digital Rights Management. TrustZone is available for A "application" and M "microcontroller" CPUs. This form of secure CPU, Trusted OS, and RoT is called a **Trusted Execution Environment (TEE)**.

- **Intel Boot Guard:** This is a hardware-based mechanism that provides a verified boot which cryptographically verifies the initial boot block or uses a measuring process for validation. Boot Guard requires a manufacturer to generate a 2048-bit key for verifying the initial block. The key is split into a private and public portion. The public key is imprinted by programmatically "blowing" fuse-bits during manufacturing. These are one-time fuses and immutable. The private portion generates the signature of the subsequently verified portion of the boot phase.

Key management and trusted platform modules

Public and private keys are critical to ensuring a secure system. The keys themselves need proper management to ensure their safety. There are hardware standards for key security and one particularly popular mechanism is the **Trusted Platform Module (TPM)**. The specification for TPM was written by the Trusted Computing Group and is an ISO and IEC standard. The current specification is TPM 2.0 released in September of 2016. Computer assets sold to the DoD require TPM 1.2.

A TPM is a discrete hardware component with a secret RSA key burned into the device at manufacturing.

Generally, a TPM is used to hold, secure, and manage other keys for services such as disk encryption, Root of Trust booting, verifying the authenticity of hardware (as well as software), and password management. A TPM can create a hash of the litany of software and hardware in a "known good" configuration that can be used to verify tampering at runtime. Additional services include assisting in SHA-1 and SHA-256 hashing, AES encryption blocks, asymmetric encryption, and random number generation. Several vendors produce TPM devices such as Broadcom, Nation Semiconductor, and Texas Instruments.

Processor and memory space

We have already discussed various exploits and processor technologies that act as countermeasures. Two predominant technologies to look for in CPU and OS facilities include non-execution memory and address space layout randomization. Both types of technologies are meant to burden or prevent buffer-overflow and stack-overrun types of malware injection:

- **Non-execution or executable space protection:** This is a facility enabled by the hardware used by the operating system to mark areas of memory as non-executable. The intent is to map only areas where verified and legitimate code resides to be the only regions of addressable memory that can execute an operation. If an attempt is made to implant malware through a stack-overflow type of attack, the stack will be marked as non-executing and an attempt to force the instruction pointer to execute there would result in a machine exception. Non-executable memory uses an NX bit as a means to map the region as non-executable (through the translation lookaside buffer). Intel uses the XD bit (execute Disable) and ARM uses an XN bit (eXecute Never). Most operating systems such as Linux, Windows, and several RTOSs support such features.
- **Address space layout randomization:** While more of an operating system treatment of virtual memory space than a hardware feature, it is important to consider ASLR. This type of countermeasure targets buffer-overflow as well as return-to-libc attacks. These attacks are based on an attacker understanding the layout of memory and forcing calls to certain benign code and libraries. Calling these libraries becomes particularly laborious if the memory space is randomized on each boot. Linux provides the ASLR ability using the PAX and Exec Shield patches. Microsoft provides protection for heap, stack, and process blocks as well.

Storage security

Often, IoT devices will have persistent storage at the edge node or on a router/gateway. Intelligent fog nodes will require persistent storage of some kind as well. The security of data on the device is imperative to prevent malicious malware from being deployed and to protect the data in the event the IoT device is stolen. Most mass storage devices such as flash modules and rotating disks have models with encryption and security technology.

The FIPS 140-2 (Federal Information Processing Standard) is a government regulation detailing encryption and security requirements for IT devices that manage or store sensitive data. It specifies not only technical requirements but also defines policies and procedures. FIPS 140-2 has several levels of compliance:



Level 1: Software-only encryption. Limited security.

Level 2: Role-based authentication is necessary. Requires the ability to detect physical tampering using tamper-evident seals.

Level 3: Includes physical tamper resistance. If the device is tampered with, it will erase critical security parameters. Includes cryptographic protection and key management. Includes identity-based authentication.

Level 4: Advanced tamper protection for products designed to work in physically unprotected environments.

In addition to encryption, it is also necessary to consider the security of media when it's decommissioned or disposed of. It is fairly easy to retrieve contents from old storage systems. There are additional standards on how to wipe and erase content securely from media (whether it's a magnetically based disk or a phase-change flash component). NIST also publishes documents on securely erasing and wiping content such as the NIST Special Publication 800-88 for Secure Erase.

Physical security

Tamper resistance and physical security are particularly important for IoT devices. In many scenarios, an IoT device will be remote and without the safeguards of on-premises devices. This is analogous to the Enigma machine of World War Two. Retrieving a working machine from the German submarine U-110 helped break the cipher. An attacker with access to the IoT device can use whatever tools are at their leisure to crack the system, as we saw with the Chain Reaction exploit.

Side channel attacks, as shown, deal with power analysis; other forms are timing attacks, cache attacks, electromagnetic field emissions, and scan chain attacks. The common theme of a side-channel attack is the compromised unit is essentially a **device under test (DUT)**. This means the device will be observed and measured in a controlled environment.

Additionally, techniques like DPA use statistical analysis approaches to look for the correlations of random input to output. The statistical analysis only works if the system behaves identically from run to run with the same input:

	Methodology
Timing Attacks	Attempts to exploit small differences in the timing of algorithms. For example, measuring the timing of a password decoding algorithm and observing early exits from the routine. Attackers also can observe cache utilization to witness the characteristics of the algorithm.
Simple Power Analysis (SPA)	Similar to a timing attack but measures large changes in dynamic power or current due to the behavior of an algorithm and opcodes. Public keys are particularly susceptible. The analysis needs few traces to work but the traces need a high degree of precision. As most cryptographic algorithms are mathematically intensive, different opcodes will show up as different power signatures in a trace.
Differential Power Analysis (DPA)	DPA measures dynamic power but can observe changes that are too small to be observed directly as in SPA. By injecting random input (such as different random keys) into a system, the attacker can perform thousands of traces to build a data-dependent set. Attacking an AES algorithm, for example, simply means building two sets of traces depending on the value of the bit (0 or 1) being cracked. The sets are averaged and the difference between the 0 and 1 set is plotted to show the effect of the random input to the output.

Methods of prevention are well known and several can be licensed and used in a variety of hardware. Countermeasures for these types of attacks include:

- Modify the encryption function to minimize the use of the key. Use a short-lived session key based on a hash of the actual key.
- For timing attacks, randomly insert functions that will not perturb the original algorithm. Use different random opcodes to create a large work function for the attacks.
- Remove conditional branches that depend on the key.
- For power attacks, decrease leakage at every opportunity and limit the number of operations per key. This reduces the attacker working set.
- Induce noise into power lines. Use variable timing operations or skew clocks.
- Change the order of independent operations. This reduces correlation factors around the S-Box calculation.

Other hardware considerations include:



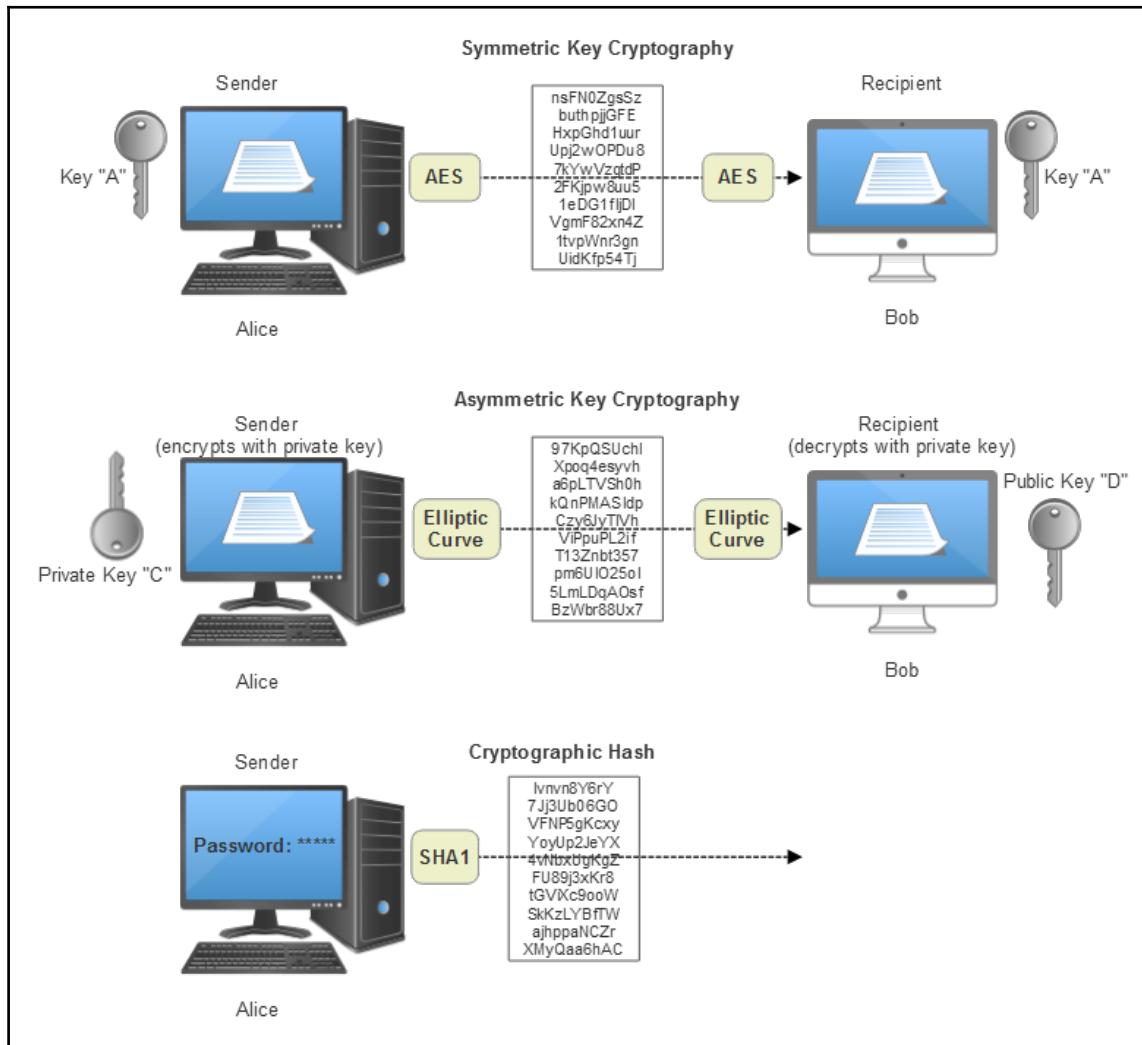
Preventing access to debug ports and channels. Often these are exposed on the PCA as serial ports and JTAG ports. Headers should be removed and fuse bits blown to prevent debug access in the most hardened cases.

ASICs typically use ball grid array (BGA) pads to attach to a PCA. High-performance adhesives and thermally-resistant glue should be used to surround the package and may cause irreparable damage if tampered with.

Cryptography

Encryption and secrecy are absolute requirements of IoT deployments. They are used for securing communication, protecting firmware, and authentication. Regarding encryption, there are generally three forms to consider:

- **Symmetric key encryption:** Encryption and decryption keys are identical. RC5, DES, 3DES, and AES are all forms of symmetric key encryption.
- **Public Key encryption:** Encryption key is published publicly for anyone to use and encrypt data. Only the receiving party has a private key used to decrypt the message. This is also known as asymmetric encryption. Asymmetric cryptography manages data secrecy, authenticates participants, and forces non-repudiation. Well-known internet encryption and message protocols such as Elliptic Curve, PGP, RSA, TLS, and S/MIME are considered public keys.
- **Cryptographic hash:** Maps data of an arbitrary size to a bit string (called the digest). This hash function is designed to be "one way". Essentially, the only way to recreate the output hash is to force every possible input combination (it cannot be run in reverse). MD5, SHA1, SHA2, and SHA3 are all forms of one-way hashes. These are typically used to encode digital signatures such as signed firmware images, **message authentication codes (MAC)**, or authentication. When encrypting a short message like a password, the input may be too small to effectively create a fair hash; in that case, a *salt* or non-private string is appended to the password to increase the entropy. A salt is a form of **key derivation function (KDF)**:



Elements of cryptography. Here are the symmetric, asymmetric, and hashing functions. Note the key usage in symmetric and asymmetric cryptography. Symmetric has the requirement of using identical keys to encrypt and decrypt data. While faster than asymmetric encryption, the keys need to be secured.

Symmetric cryptography

In encryption, plaintext refers to the unencrypted input and the output is called ciphertext, as it is encrypted. The standard for encryption is the **Advanced Encryption Standard (AES)** which replaced older DES algorithms dating from the 1970s. AES is part of the FIPS specification and the ISO/IEC 18033-3 standard used worldwide. AES algorithms use fixed blocks of 128, 192, or 256 bits. Messages larger than the bit width will be split into multiple blocks. AES has four basic phases of operation during the cipher. The pseudo code for a generic AES encryption is shown here:

```
// Psuedo code for an AES-128 Cipher
// in: 128 bits (plaintext)
// out: 128 bits (ciphertext)
// w: 44 words, 32 bits each (expanded key)
state = in

w=KeyExpansion(key) //Key Expansion phase (effectively encrypts key itself)
AddRoundKey(state, w[0, Nb-1]) //Initial Round

for round = 1 step 1 to Nr-1 //128 bit= 10 rounds, 192 bit = 12 rounds, 256
bit = 14 rounds
    SubBytes(state) //Provide non-linearity in the cipher
    ShiftRows(state) //Avois columns being encrypted independently which can
weaken the algorithm
    MixColumns(state) //Transforms each column and adds diffusion to the
cipher
    AddRoundKey(state, w[round*Nb, (round+1)*Nb-1]) //Generates a subkey an
combines it with state.
end for

SubBytes(state) //Final round and cleanup.
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

out = state
```



AES key lengths can be 128, 192, or 256 bits. Generally, the larger the key length the better the protection. The size of the key is proportional to the number of CPU cycles needed to encrypt or decrypt a block: 128 bit needs 10 cycles, 192 bit needs 12 cycles, and 256 bit needs 14 cycles.

Block ciphers represent encryption algorithms that are based on a symmetric key and operate on a single block of data. Modern ciphers are based on Claude Shannon's work on product ciphers in 1949. A cipher mode of operation is an algorithm that uses a block cipher and describes how to repeatedly apply a cipher to transform large amounts of data consisting of many blocks. Most modern ciphers also require an **Initialization Vector (IV)** to ensure distinct ciphertexts even if the same plaintext is entered repeatedly. There are several modes of operation such as:

- **Electronic Codebook (ECB):** This is the most basic form of AES encryption, but it is used with other modes to build more advanced security. Data is divided into blocks and each is encrypted individually. Identical blocks will produce identical ciphers which makes this mode relatively weak.
- **Cipher Block Chaining (CBC):** Plaintext messages Xored with the previous ciphertext before being encrypted.
- **Cipher Feedback Chaining (CFB):** Similar to CBC but forms a stream of ciphers (the output of the previous cipher feeds into the next). CFB depends on the previous block cipher to provide input to the current cipher being generated. Because of the dependency of previous ciphers, CFB cannot be processed in parallel. Streaming ciphers allows for a block to be lost in transit but subsequent blocks can recover from the damage.
- **Output Feedback Chaining (OFB):** Similar to CFB, this is a streaming cipher but allows for error correcting codes to be applied before encryption.
- **Counter (CTR):** Turns a block cipher into a stream cipher but uses a counter. The incrementing counter feeds each block cipher in parallel allowing for fast execution. The nonce and counter are concatenated together to feed the block cipher.
- **CBC with Message Authentication Code (CBC-MAC):** A MAC (also known as tag or MIC) is used to authenticate a message and confirm the message came from the stated sender. The MAC or MIC is then added to the message for verification by the receiver.

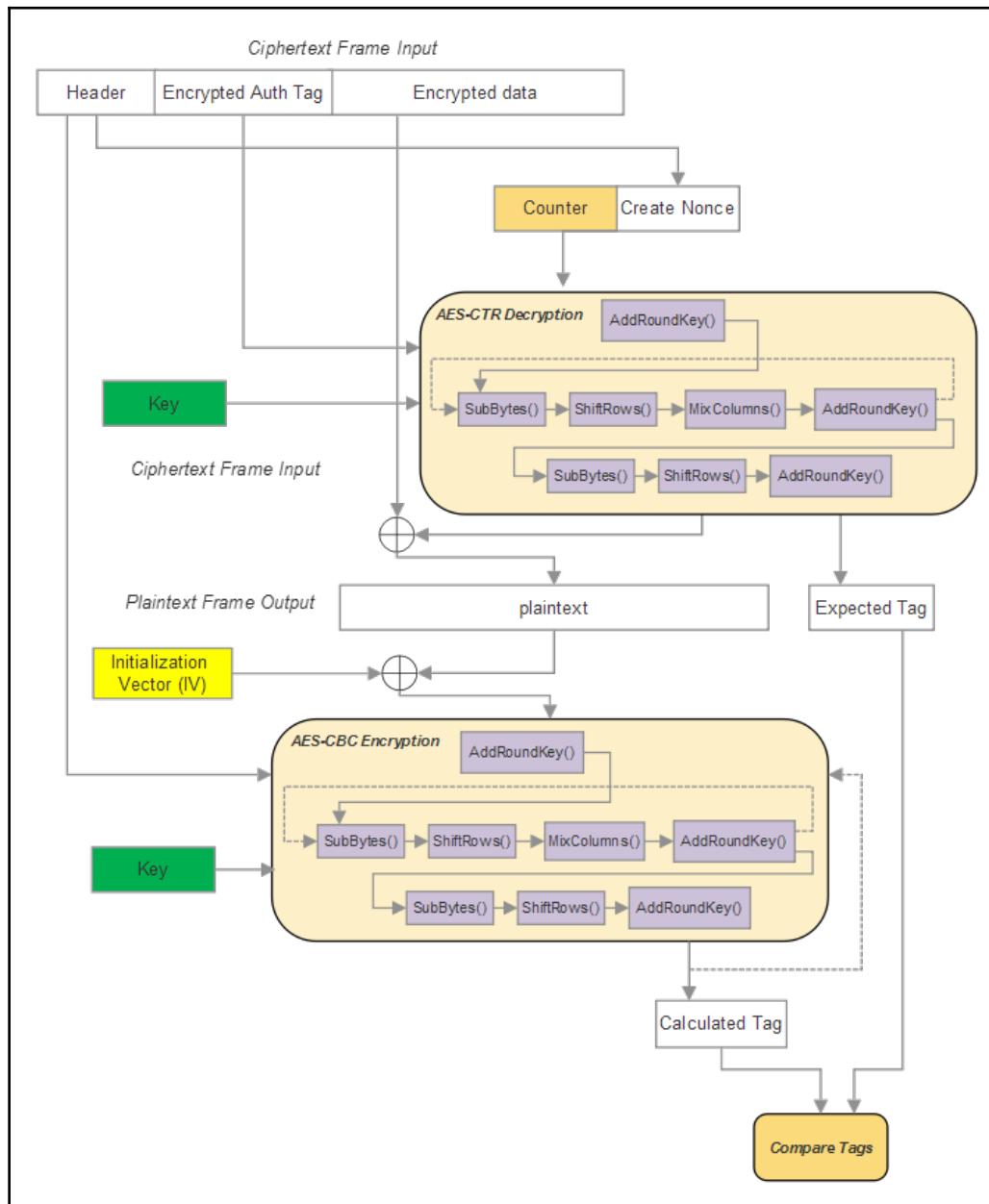
These modes were first constructed in the late 1970s and early 1980s and were advocated by the National Institute of Standards and Technology in FIPS 81 as DES modes. These modes provide encryption for the confidentiality of information but will not protect against modification or tampering. To do that, a digital signature is needed and the security community developed CBC-MAC for authentication. Combining CBC-MAC with one of the legacy modes was difficult until algorithms like AES-CCM were established, which provide both authentication as well as secrecy. CCM stands for Counter with CBC-MAC Mode.



CCM is an important encryption mode used to sign and encrypt data and is used in a plethora of protocols covered in this book including Zigbee, Bluetooth Low Energy, TLS 1.2 (after key exchange), IPSEC, and 802.11 Wi-Fi WPA2.

AES-CCM makes use of dual ciphers: CBC and CTR. AES-CTR or counter mode is used for the general decryption of the ciphertext stream flowing in. The incoming stream contains an encrypted authentication tag. AES-CTR will decrypt the tag as well as the payload data. An "expected tag" is formed from this phase of the algorithm. The AES-CBC phase of the algorithm tags as input the decrypted blocks from the AES-CTR output and the original header of the frame. The data is encrypted; however, the only relevant data needed for authentication is the calculated tag. If the AES-CBC calculated tag differs from the AES-CTR expected tag, then there is the possibility that the data was tampered with in transit.

The figure below illustrates an incoming encrypted stream of data that is both authenticated using AES-CBC and decrypted using AES-CTR. This ensures both the secrecy and authenticity of the origin of a message:



AES-CCM mode.



One consideration for IoT deployments in a fully connected mesh is the number of keys necessary. For n nodes in a mesh that desire bidirectional communication, there are $n(n-1)/2$ keys or $O(n^2)$.

Asymmetric cryptography

Asymmetric cryptography is also called public key cryptography. Asymmetric keys are generated in pairs (encrypting and decrypting). The keys can be interchangeable meaning a key could both encrypt and decrypt, but that is not a requirement. The typical use, however, is to generate a pair of keys and keep one private and the other public. This section describes the three foundational public key cryptograms: RSA, Diffie-Hellman, and Elliptical Curves.



Note that unlike symmetric key counts in a mesh where any node can communicate with any other node, asymmetric cryptography only requires $2n$ keys or $O(n)$.

The first asymmetric public key encryption method described is the *Rivest-Shamir-Adleman* algorithm, or RSA, developed in 1978. It is based on a user finding and publishing the product of two large prime numbers and an auxiliary value (public key). The public key can be used by anyone to encrypt a message but the prime factors are private. The algorithm operates as follows:

1. Find two large primes, p and q
2. $n = pq$
3. $\varphi(n) = (p-1)(q-1)$
4. **Public key:** Choose an integer e so that $1 < e < \varphi(n)$, e is coprime to $\varphi(n)$; a typical value is $2^{16} + 1 = 65537$
5. **Private key:** Compute d to solve the congruence relation $de \equiv 1 \pmod{\varphi(n)}$

Therefore, to encrypt a message using the public key (n,e) and to decrypt a message using the private key (n,d) :

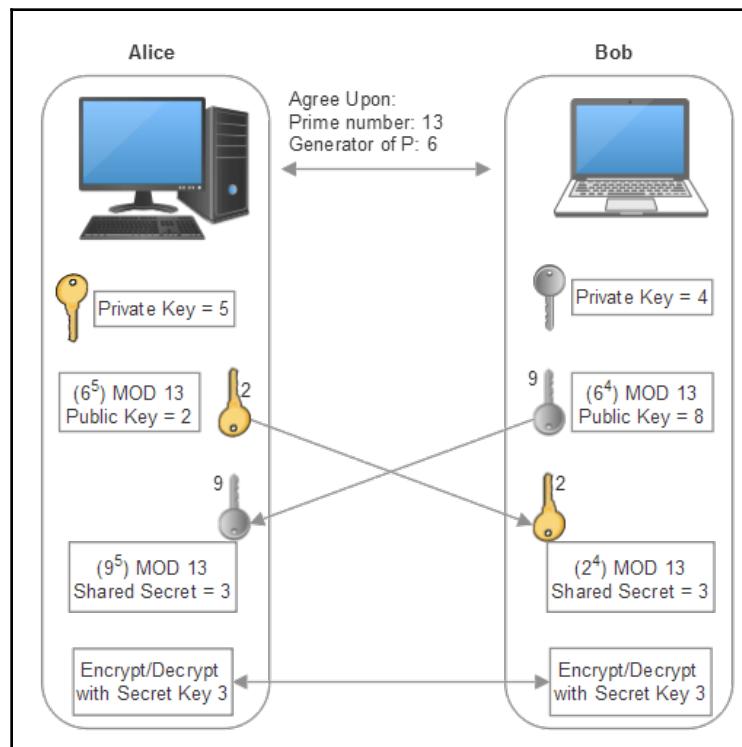
- **Encrypt:** $Ciphertext = (plaintext)^e \pmod{n}$
- **Decrypt:** $Plaintext = (ciphertext)^d \pmod{n}$

It is often the case that artificial padding is injected into m before encryption to avoid cases of short messages failing to produce good ciphertext.

Perhaps the best-known form of asymmetric key exchange is the Diffie-Hellman key exchange process (named after Whitfield Diffie and Martin Hellman). Typical of asymmetric cryptography is the notion of a *Trapdoor Function* which takes a given value A and produces an output B. However, trapdoor function B does not produce A.

The *Diffie-Hellman* method of key exchange allows two parties (Alice A and Bob B) to exchange keys without any previous knowledge of any shared secret key s . The algorithm is based on the plaintext exchange of a starting prime number, p , and a prime generator. The generator g is a primitive root modulo p . Let Alice's private key be a and Bob's private key be b . Then, $A=g^a \text{ mod } p$ and $B=g^b \text{ mod } p$. Alice computes the secret key as $s=B^a \text{ mod } p$ and Bob computes the secret key as $s=A^b \text{ mod } p$.

In general, $(g^a \text{ mod } p)^b \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p$:



Diffie-Hellman key exchange. The process begins with the plaintext exchange of an agreed upon prime and generator of prime. With an independent private key generated by Alice and Bob, the public keys are generated and sent in plaintext over the network. This is used to generate a secret key used for encryption and decryption.

The strength of this form of secure key exchange is generating a true random number for each private key. Even the slight predictability of a **pseudo-random number generator (PRNG)** can lead to breaking the encryption. However, the principal issue is the lack of authentication which can lead to an MITM attack.

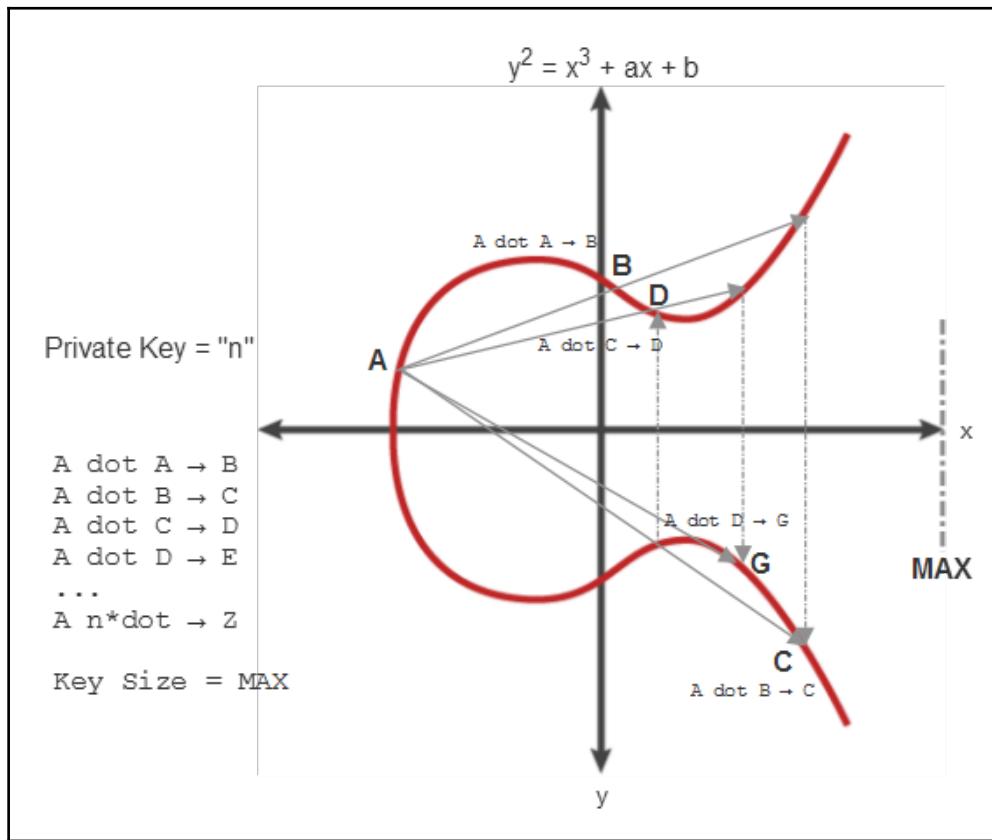
The other form of key exchange is *Elliptic-Curve Diffie-Hellman* (ECDH) and was proposed by Koblitz and Miller in 1985. It is based on the algebra of elliptic curves over a finite field. NIST has endorsed ECDH and the NSA allows ECDH for top-secret material using 384-bit keys. **Elliptic Curve Cryptography (ECC)** shares these basic tenets regarding the properties of an elliptical curve:

- Symmetric about the x-axis
- A straight line will intersect the elliptic curve at no more than three points

The process of ECC starts with drawing a straight line from a given point on the edge towards MAX. A line is drawn from point *A* to *B*. A dot function is used to draw a line between two points and then to draw a line straight up (or down) from the new unlabelled intersection to its counterpart on the positive or negative y-axis. This process is repeated *n* times, where *n* is the key size.

This process is analogous to seeing the end results of a pool table after a ball has been hit and strikes the cushions many times. An observer seeing the ending location of the ball would have a difficult time determining where the original location of the ball was.

MAX corresponds maximum value on the x-axis, placing a limit on how far a vertex will stretch. If by chance the vertex is greater than MAX, the algorithm forces the value that would have extended beyond the MAX limit and sets a new point *x-MAX* distance away from the origin *A*. MAX is equivalent to the key size being used. A large key size results in more vertices being constructed and increasing strength of secrecy. Essentially this is a wrap around function:



Elliptical Curve Cryptography (ECC). Here is a standard elliptical curve on an x, y-axis. The process starts by following straight line paths from a given point A to a second point and locating the third new unlabelled intersection. A line is drawn to the opposite but identical y plane coordinate which now becomes a labeled entity. The process continues for n points which correspond to the key length.

Elliptical curves are becoming prevalent over RSA. Modern browsers are capable of supporting ECDH, which is the preferred method of authentication over SLL/TLS. ECDH is also found in Bitcoins, as we will see later, and several other protocols. RSA is only used now if an SSL certificate has a matching RSA key.



The other advantage is that the key lengths can be kept short and still have the same cryptographic strength as a legacy method. For example, a 256-bit key in ECC is equivalent to a 3072-bit key in RSA. The importance of this should be considered for constrained IoT devices.

Cryptographic hash (authentication and signing)

Hashing functions represent the third type of cipher technology to consider. These are typically used to generate a digital signature. They are also considered "one-way" or impossible to invert. To recreate the original data after passing through a hash function would be a brute-force attack of every possible combination of inputs. The key attributes of a hashing function include:

- Always generates the same hash from the same input
- Quick to compute but not free (see proof of work)
- It is non-reversible and cannot regenerate the original message from the hash value
- A small change to the input will result in significant entropy or a change in the output
- Two different messages will never have the same hash value

The effects of cryptographic hashing functions like SHA1 (Secure Hash Algorithm) can be demonstrated by altering just one character in a longer string:



Input: Boise Idaho

SHA1 Hash Output: 375941d3fb91836fb7c76e811d527d6c0a251ed4

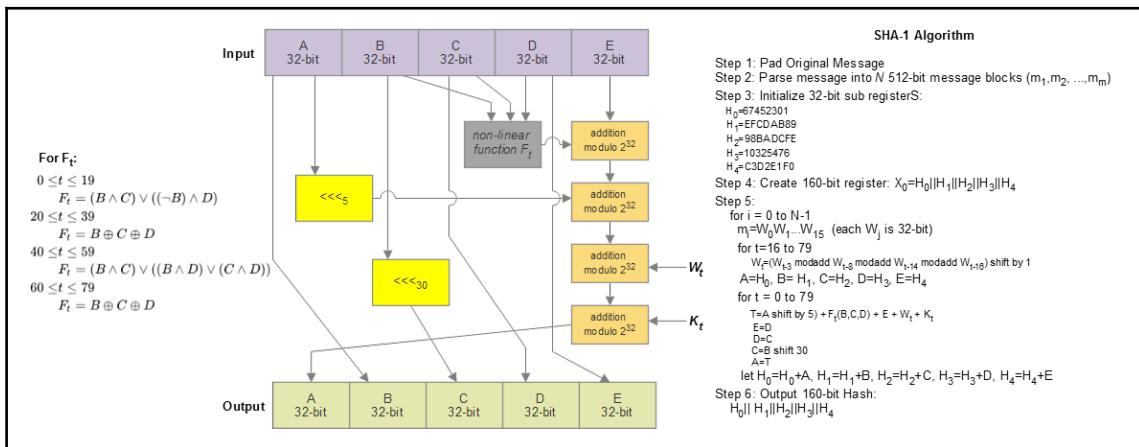
Input: Boise Idaho

SHA1 Hash Output: 82b6109838f8f40dc1d1530e5535908853e3fd5f

SHA algorithms are used extensively in applications such as:

- Git repositories
- TLS certificate signing for web browsing (HTTPS)
- Validating file or disk image content authenticity

Most hash functions are built upon the Merkle-Damgård construction. Here, the input is split into equally sized blocks which are processed serially with a compression function combined with the output of the previous compression. An **Initialization Vector (IV)** is used to seed the process. By using a compression function, the hash is resistant to collisions. SHA-1 is built upon this Merkle-Damgård construction:



SHA-1 algorithm. Input split into five 32-bit chunks.

In general, the SHA algorithm's input message must be less than 264 bits. The message is processed in 512-bit blocks sequentially. SHA-1 is now superseded with strong kernels such as SHA-256 and SHA-3. SHA-1 was found to have "collision" within the hash. While it would take approximately 2^{51} to 2^{57} operations to find a collision, it would take only a few thousand dollars of rented GPU time to resolve the hash. Thus, the recommendation is to move to the strong SHA models.

Public Key Infrastructure

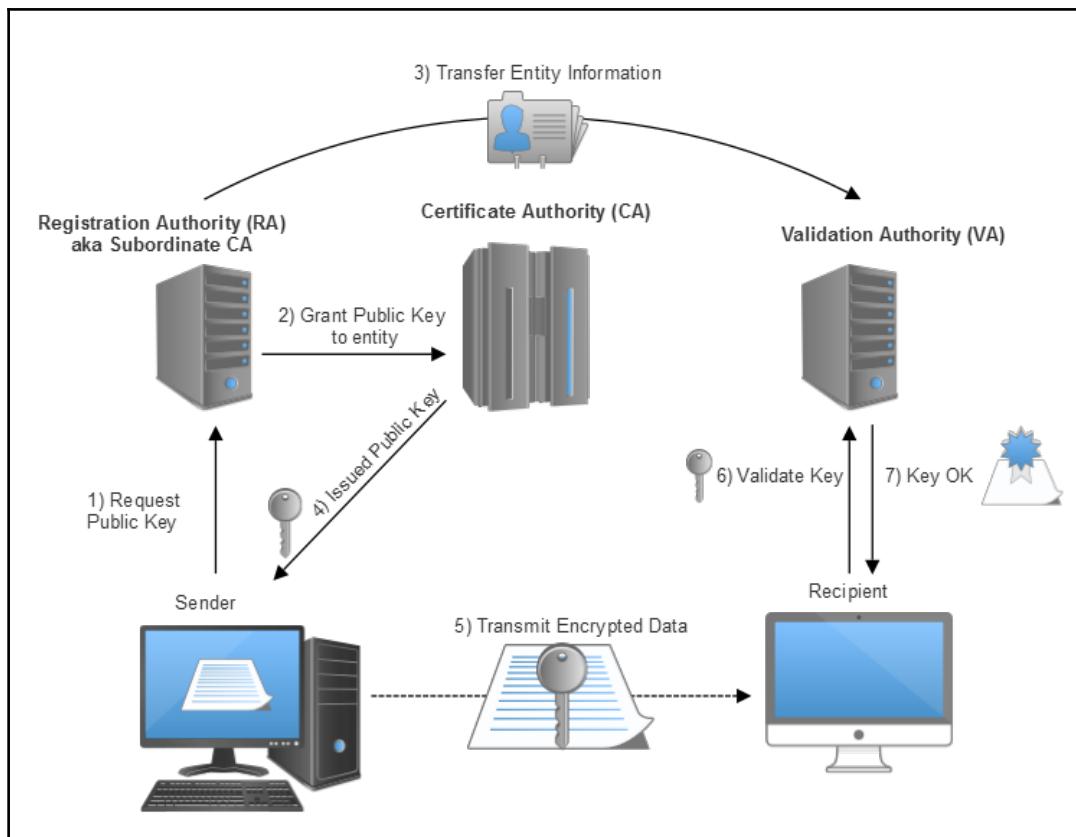
Asymmetric cryptography (public key) is a mainstay of internet commerce and communication. It has routinely been used for SSL and TLS connection on the web. A typical use is public key encryption, where data in transit is encrypted by anyone holding the public key but can only be decrypted by the holder of the private key. Another use is digital signatures, where a blob of data is signed with a sender's private key and the receiving party can verify the authenticity if they hold the public key.

To assist with providing public keys with confidence, a process known as **Public Key Infrastructure (PKI)** is used. To guarantee authenticity, trusted third parties known as **Certificate Authorities (CA)** manage roles and policies to create and distribute digital certificates. Symantec, Comodo, and GoDaddy are the largest public issuers of TLS certificates. X.509 is a standard that defines public key certificate formats. It is the basis of TLS/SSL and HTTPS secure communication. X.509 defines such things as the encryption algorithm used, expiration dates, and the issuer of the certificate.



PKI consists of **Registration Authority (RA)** that verifies the sender and manages specific roles and policies and can revoke certificates. The RA also communicates with a **Validation Authority (VA)** to transfer revocation lists. The CA issues the certificate to the sender. When a message is received, the key can be validated by the VA to verify that it hasn't been revoked.

In the following figure, an example of PKI infrastructure is shown. The CA, RA, and VA systems used and the phases of granting and verifying a key granted for encrypting a message are shown:



PKI infrastructure example.

Network stack – Transport Layer Security

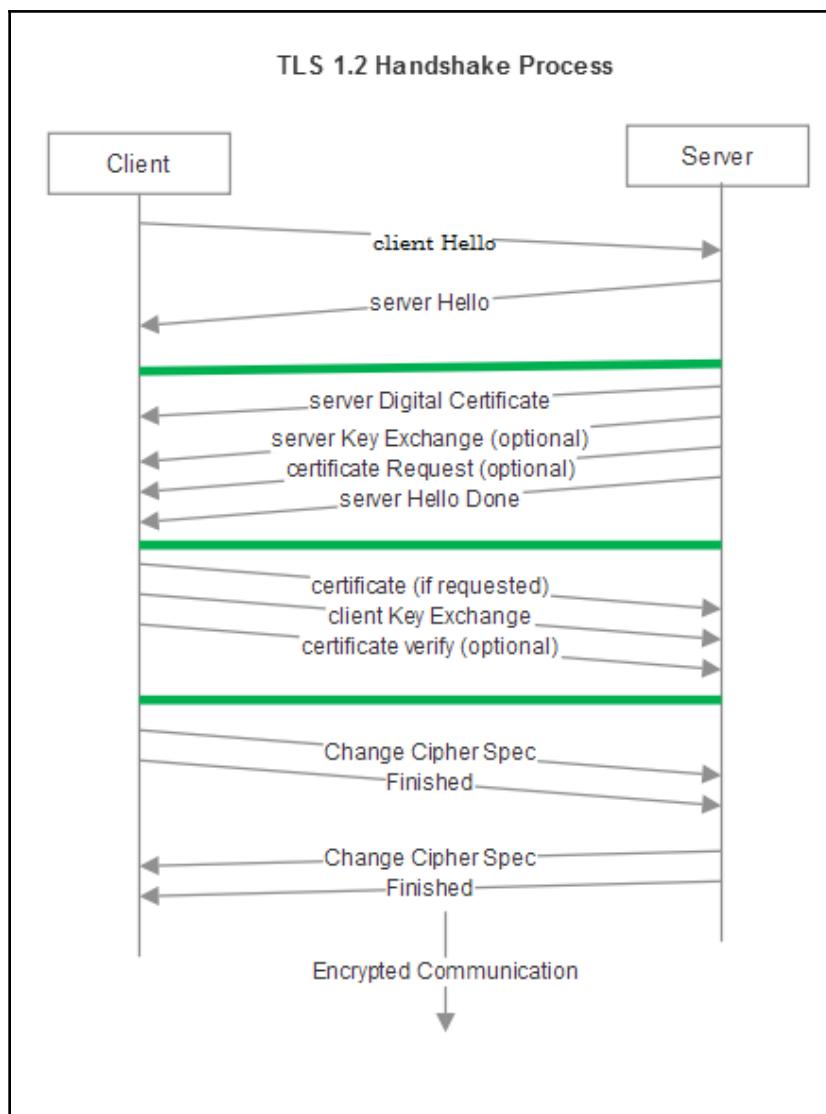
Transport Layer Security (TLS) has been covered in many areas of this book, from TLS and DTLS for MQTT and CoAP to network security over the WAN and PAN security. Each has had some form of reliance on TLS. TLS also brings together all the cryptographic protocols and technologies we mention. This section briefly covers the TLS1.2 technology and process.

Originally SSL (Secure Sockets Layer), it was introduced in the 1990s but was replaced by TLS in 1999. TLS 1.2 is the current specification in RFC5246 as of 2008. TLS 1.2 includes an SHA-256 hash generator to replace SHA-1 and to strengthen its security profile.

In TLS encryption, the process is as follows:

1. The client opens a connection to a TLS-capable server (port 443 for HTTPS)
2. The client presents a list of supported ciphers that can be used
3. Service picks a cipher and hash function and notifies the client
4. The server sends a digital certificate to the client which includes a certificate authority and the server's public key
5. Client confirms the validity of the secret
6. A session key is generated either by:
 1. A random number being encrypted with the server's public key, and sending the result to the server. Server and client then use the random number to create a session key which is used for the duration of the communication.
 2. Using Dixie-Hellman key exchange to generate a session key used for encryption and decryption. The session key is used until the connection is closed.
7. Communication starts using the encrypted channel

Shown below is the handshaking process for TLS1.2 communication between two devices:



Datagram Transport Layer Security (DTLS) is a communications protocol at the datagram layer based on TLS (DTLS 1.2 is based on TLS 1.2). It is intended to produce similar security guarantees. The CoAP lightweight protocol uses DTLS for security.

Software defined perimeter

Earlier in the book, in Chapter 8, *Router and Gateways*, the concept of software-defined networks and overlay networks was discussed. Overlay networks and their ability to create microsegments is extremely powerful especially in mass IoT scaling and situations where a DDOS attack can be mitigated. An additional component of software-defined networking is called **Software-Defined Perimeter (SDP)** and is worth a discussion in terms of the overall security picture.

Software-Defined Perimeter architecture

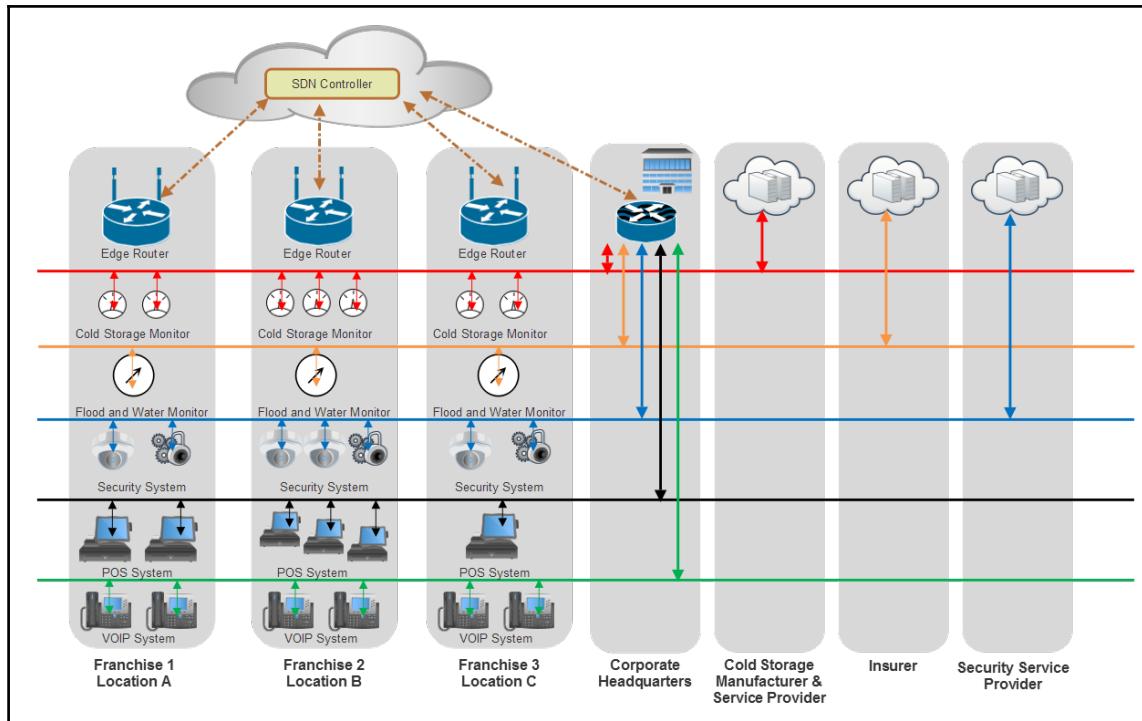
A **Software-Defined Perimeter (SDP)** is an approach to network and communication security where no trust model exists. It is based on the **Defense Information Systems Agency (DISA)**'s black cloud. Black cloud means information is shared on a need-to-know basis. An SDP can mitigate attacks such as DDOS, MITM, zero-day exploits, and server scanning among others. Along with providing an overlay and micro-segmentation for each attached device, the perimeter creates an invitation-only (identity-based) security perimeter around users, clients, and IoT devices.

An SDP can be used to create an overlay network which is a network, built on top of another network. A historical reference is legacy internet services built upon the pre-existing telephone network. In this hybrid networking approach, the distributed control plane remains the same. Edge routers and virtual switches steer data based on control plane rules. Multiple overlay networks can be built on the same infrastructure. Since an SDN remains persistent in much the same way as a wired network, it is ideal for real-time applications, remote monitoring, and complex event processing. The ability to create multiple overlay networks using the same edge components allows for *micro-segmentation* where different resources have a direct relationship with different consumers of data. Each resource-consumer pair is a separate immutable network that is only able to see outside of its virtual overlay as the administrator chooses.



The ability to create multiple overlay networks using the same edge components allows for micro-segmentation where each endpoint in a globally distributed IoT network can build individual and isolated network segments over the existing network infrastructure. In theory, every sensor could be isolated from another. This is a powerful tool for enabling enterprise connectivity for IoT deployments as services and devices can be isolated and protected from each other.

The following graphic depicts an SDN overlay example. Here, a corporation has three remote franchises with a number of different IoT and edge devices at each store. The network resides on an SDN overlay network with micro-segments isolating POS and VOIP systems that are corporately managed from various sensors for security, insurance, and cold storage monitoring. Third-party service providers can manage various remote sensors using an isolated and secure virtual overlay network only for the device they manage:



An example SDN Overlay Network.

An SDP can further extend security by developing an invitation system, forcing a pair of devices to authenticate first and connect second. Only a pre-authorized user or client may be added to a network. That authorization is extended by an invitation from the control plane via email or some registration facility. If the user accepts the invitation, client certificates and credentials are extended to that system alone. The resource extending the invitation maintains a record of extended certificates and will only provide an overlay connection when both parties accept the role.

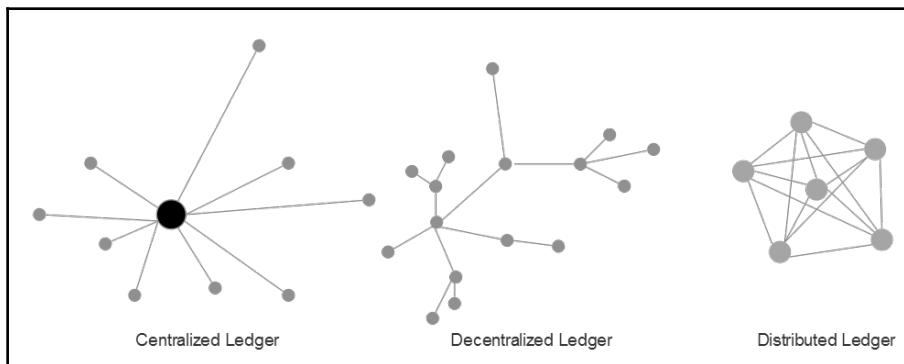


The real-world analogy is how one sends invitations for a party. Invitations are mailed out to selected individuals, with a date, time, address, and other details. Those individuals may or may not want to attend at the party (it is up to them). The alternative is to advertise on the web, TV, and radio that you are having a party and then authenticate each person as they arrive at the door.

Blockchains and cryptocurrencies in IoT

Blockchains are public, digital, and decentralized ledgers or cryptocurrency transactions. The original cryptocurrency blockchain was Bitcoin but there are over 700 new currencies on the market such as Ethereum, Ripple, and Dash. The power of a blockchain is that there is no single entity controlling the state of transactions. It also forces redundancy in the system by ensuring everyone using a blockchain also maintains a copy of the ledger. Assuming there is no inherent trust in blockchain participants, the system must live in consensus.

A good question to ask is if we have solved identity management and security with asymmetric cryptography and key exchanges, why are blockchains needed to exchange data or currency? This is not enough for the exchange of money or data of value. One thing to note is that since the inception of information theory, when two devices communicate, Bob and Alice, they send a message or a bit of data. That information is still retained by Bob even if Alice receives a copy. When exchanging money or contracts, that data must leave one source and arrive at another. There is only a single instance. Authenticity and encryption are tools needed for communication but a new method to transfer ownership had to be invented:



Ledger topologies. A centralized ledger is the typical process of a single controlling agent maintaining "the books". Cryptocurrencies use either decentralized or distributed ledgers

Blockchain secure cryptocurrencies are particularly relevant to IoT. Some example use cases include:

- **Machine to machine payments:** The IoT needs to ready itself for machine exchanging services for currency.
- **Supply chain management:** In this case, the movement and logistics in managing inventory, moving goods, and logistics can replace paper-based tracking with blockchain immutability and security. Every container, movement, location, and state can be tracked, verified, and certified. Attempts to forge, delete, or modify tracking information becomes impossible.
- **Solar energy:** Imagine residential solar as a service. In this case, solar panels installed on a customer home are generating energy for the home. Alternatively, they can also send energy back to the grid to power someone else (perhaps in exchange for carbon credits).

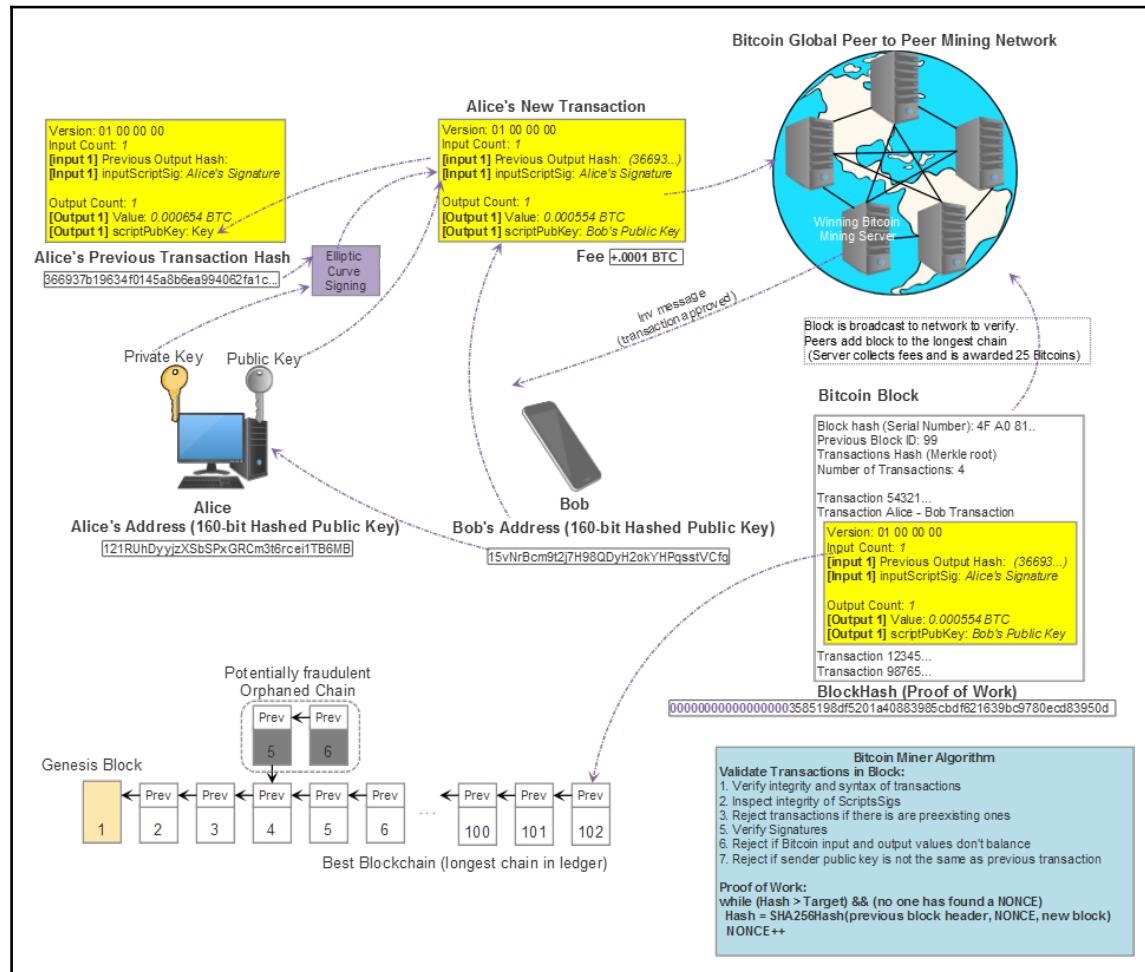
Bitcoin (blockchain-based)

The cryptocurrency portion of Bitcoin is different from the blockchain itself. Bitcoin is an artificial currency. It has no commodity or value backing like gold. It is also not physical; it only exists in a network construct. Finally, the supply or number of Bitcoins is not determined by a central bank or any authority. It is completely decentralized. Like other blockchains, it is built up from public key cryptography, a large and distributed peer-to-peer network, and a protocol that defines the Bitcoin structure. While not the first to conceive of digital cash, Satoshi Nakamoto (alias) posted the paper in 2008 called *Bitcoin: A Peer-to-Peer Electronic Cash System* to a cryptography list. In 2009, the first Bitcoin network was online and Satoshi mined the first block (Genesis Block).

The concept of a blockchain means that there exists a *block* representing the current portion of a blockchain. A computer connected to the blockchain network is called a *node*. Each node participates in validating and relaying transactions by obtaining a copy of the blockchain and is essentially an administrator.

A distributed network based on peer-to-peer topologies exists for Bitcoin. Metcalfe's Law applies here since the value of a cryptocurrency like Bitcoin is based on the size of the network. The network maintains the system of records (the ledger). The problem is, where do you find a source of computing willing to share their compute time for monitoring ledgers? The answer is to build a reward system called *Bitcoin mining*.

The transaction process is shown in the following figure. It starts with a transaction request. The request is broadcast to a Peer-to-Peer (P2P) network of computers called **nodes**. The peer network is responsible for validating the authenticity of the user. Upon validation, the transaction is verified and combined with other transactions to create a new block of data for the distributed ledger. When a block is full, it is added to the existing blockchain, making it immutable. The Bitcoin authentication, mining, and validation process is shown below:



Bitcoin blockchain transaction process.

Shown in the graphic is the exchange of 0.000554 Bitcoins between Alice and Bob with a service fee of 0.0001 BTC. Alice initiates the transaction by signing the contents of the transaction with the previous transaction hash against Alice's private key. Alice also includes her public key in the inputScriptSig script. The transaction is then broadcast to the Bitcoin P2P network for inclusion in a block and validation. The network competes to validate and discover a working nonce based on the current strength of complexity. If a block is discovered, the server broadcasts the block to peers for validation and then inclusion in the chain.

What follows is a qualitative analysis of the blockchain and in particular Bitcoin processing. It is important to understand these fundamentals which build on all the security foundations from earlier in this chapter:

1. **Digitally signed transaction:** Alice intends to give Bob 1 Bitcoin. The first step is to announce to the world that Alice intends to give Bob 1 Bitcoin. Alice does this by writing this message: "Alice will give Bob 1 Bitcoin," and digitally signing it for authentication with her private key. Anyone can verify the message is authentic given the public key. However, Alice could replay the message and artificially forge money.
2. **Unique identification:** To resolve the forgery problem, Bitcoin creates a unique with a serial number. US-issued money has serial numbers and Bitcoins do as well in a general sense. Bitcoin uses a hash rather than a centrally administered serial number. The hash that identifies the transactions are self-generated as part of a transaction.
A serious problem arises with double spending. Even if the transaction is signed and uniquely hashed, Alice could potentially reuse the same Bitcoin with other parties. Bob will check Alice's transaction and everything will verify. If Alice also uses the same transaction but buys something from Charlie, she effectively is cheating the system. The Bitcoin network is very large, but there is still a small chance that theft could occur. Bitcoin users protect against double spending by waiting for a confirmation when receiving payments on the blockchain. As a transaction becomes dated more confirmations arise, validating it becomes more irreversible.
3. **Security through peer validation:** To resolve the double spending cheat, what is done in blockchains is that recipients of the transaction (Bob and Charlie) broadcast their potential payment to the network and request the peer network to help legitimize it. This service of requesting assistance in verifying a transaction doesn't come free.

4. **Proof of Work Burden:** This still hasn't completely resolved the double spend problem. Alice could simply hijack the network with her own servers and claim all her transactions are valid. To finally resolve this issue, Bitcoin introduced the concept of *Proof of Work*. There are two aspects to proof-of-work concepts. The first aspect is validating the authenticity of a transaction should be computationally expensive for a computing device. It must be more computationally burdensome than just verifying keys, logins names, transaction IDs, and other trivial steps in the authentication process. Second, users need to be rewarded to help resolve other people's money exchanges—this is covered in step
5. The method Bitcoin uses to force a work function on individuals validating transactions is to attach a nonce to the header of the transactions in process. Bitcoin then uses the cryptographically secure SHA-256 algorithm to hash the nonce and header message. The goal is to keep changing the nonce and deliver hash leading values that are less than 256-bit values, known as the **target**. A low target makes it much more computationally intensive to resolve. Since each hash basically generates a completely random number, many SHA-256 hashes must be performed. On average this takes about 10 minutes to resolve.



The 10-minute proof-of-work also implies that a transaction will take 10 minutes on average to validate. Miners work on blocks which are collections of many transactions. A block is limited (currently) to 1 MB of transactions which implies your transaction will not be processed until the current block is completed. This can have implications for an IoT device with real-time demands.

6. **Bitcoin Mining Incentives:** To encourage individuals to build a peer-to-peer network to validate other people's spending, incentives are used to reward those individuals for their service. There are two forms of an reward. The first is Bitcoin Mining, which awards individuals who validate a block of transactions. The other form of award is a transaction fee. The transaction fee is rewarded to a miner who helps validate a block and is taken as part of the transaction. Initially, there was no fee, but as Bitcoins have gained popularity, the fee has grown. On average the fee is about \$35 (in Bitcoin currency) for a successful transaction. As a further incentive, fees are dynamic and one can increase fees, which artificially forces the transaction to be processed faster for a user. Even when new Bitcoin generation is exhausted, there is still an incentive to manage transactions.

Initially, the reward was very high (50 Bitcoins) but that reduces by half about every four years after 210,000 blocks have been found. This will continue until the year 2140 when the halving rate will reach a point of exhaustion and rewards will be less than the lowest unit value of a Bitcoin (called a Satoshi or 10^{-8} Bitcoins).



Given that a block is mined every 10 minutes and the reward is halved every four years, we can come up to the maximum number of Bitcoins in existence. We also know the initial reward for a mined coin is 50 Bitcoins. This produces a series that converges to the Satoshi limit: $50 \text{ BTC} + 25 \text{ BTC} + 12.5\text{BTC}... = 100 \text{ BTC}$. $210,000 * 100 = 21 \text{ million total Bitcoins}$ possible.

7. **Security through chaining order:** The order in which transactions occur is also vital to the integrity of a currency. If a Bitcoin is transferred from Alice to Bob and then to Charlie, you don't want the ledgers to record the events as Bob to Charlie then Alice to Bob. Blockchains manage the order by *chaining* the transactions. All new blocks added to the network contain a pointer to the last block validated in the chain. Bitcoin states that no transaction is valid until it is chained to the longest fork and that at least five blocks follow it in the longest fork.
This provision solves the asynchronous problem of what happens if Alice tries to double spend a Bitcoin with Bob and Charlie. She might try to broadcast the transaction with Bob for one set of the miners and with Charlie on a second set. However, the process will find this fraud when the network converges. Bob may win the transaction but the network will invalidate Charlie's transaction.
Even if Alice attempts to pay herself and attempts to pay Bob, the ordering rules stop her. She does this by sending Bob a Bitcoin and waiting for the moment when the transaction is verified (five blocks follow it). She then immediately pays herself the same Bitcoin which causes a fork in the chain. She now has to validate five additional blocks of Bitcoins. This takes about 50 minutes as stated in step 4. That requires an enormous amount of computing power as she has to process faster than all other miners combined.



Another interesting concept about blockchains is their use in managing the denial of service attacks. A proof-of-work system (or protocol/function) is an economic measure to deter the denial of service attacks. One conducting an attack intends to saturate a network with as much data as possible to overwhelm a system. A blockchain involving a work function reduces the effectiveness of such an attack. A key feature of these schemes is their asymmetry: the work must be moderately hard (but feasible) on the requester side but easy to check with the service provider.

IOTA (directed acyclical graph-based)

An interesting new cryptocurrency has been developed solely for IoT called IOTA. In this case, the IOT devices themselves are the backbone of the trust network and the architecture is based on a **Directed Acyclic Graph (DAG)**. Bitcoin has an associated fee per transaction. IOTA has no fees. This is very important in the world of IoT, which may be serving micro-transactions. For example, a sensor may provide a service reporting to many subscribers of MQTT. The service has value in aggregate terms but is measurably very small per single transaction, so small that a Bitcoin fee to provide that information may be larger than the value of the data.

The IOTA architecture has the following aspects:

- No centralization of monetary control; with blockchain, miners can form large groups in order to increase the number of blocks they can mine as well as increase their rewards. This can potentially lead to a concentration of power and possibly harm the network.
- No costly hardware equipment. In order to mine Bitcoins, powerful processors are necessary due to the complexity of the processing logic required.
- Micro and nano transactions at the IoT level of data.
- Proven security even against quantum computing brute force attacks.
- Data that can be transferred through IOTA just as currency can. The data is fully authenticated and tamper-proof.
- Since IOTA is agnostic to the payload of a transaction, a national tamper-proof voting system can be designed.
- Anything with a small SOC can become a service. If you own a device such as a drill, personal Wi-Fi, a microwave, or bicycle, the devices with a small SOC or microcontroller can join IOTA and become leasing sources of revenue.

The IOTA DAG is called a **tangle** and is used store transactions as a distributed ledger. Transactions are issued by nodes (IoT devices) and they constitute a set on the tangle DAG. If there is no directed edge between transaction A and transaction B, but there is a path of at least length two from A to B, it can be said A indirectly approves B.

There is also the notion of a *genesis* transaction. Since there is no mining for graph edges (and no incentives or fees) to start the tangle, one node contains all tokens. The genesis event sends the tokens to other "founder" addresses. This is the static set of all tokens and no new tokens will ever be created.

When a new transaction arrives, it must approve (or deny) two previous transactions; this is called direct approval. This forms a direct edge on the graph. Anything performing a transaction is required to produce a "work" product on behalf of the tangle. This work involves finding a nonce that works with the hash of a portion of the approved transaction. Thus by using IOTA, the network becomes more distributed and more secure. The transaction may have many approvers. As more transactions occur, the confidence that the transaction is legitimate increases. If a node attempts to approve a transaction that for all intents is not legitimate, it risks having its own transaction be continuously disproved and falling into oblivion.

While still early in the rollout, it is a technology to watch. More information can be found at <http://iota.org>.

Government regulations and intervention

Government bodies and regulatory agencies have stepped in to recommend and mandate levels of security that must be met by vendors to be considered secure. There has been more interest lately in the US government with regards to ensuring certain security standards are met for connected devices, especially with the growing attacks on IoT systems. It is important to be cognizant of these rules and laws as other countries may adopt similar or, in other cases, completely dissimilar regulations making the job of an IoT architect difficult when scaling globally. These laws affect privacy and security for the individual as well as the nation.

US Congressional Bill –Internet of Things (IoT) Cybersecurity Improvement Act of 2017

IoT security has received attention from various governments. A bipartisan bill (S.1691 – Internet of Things IoT Cybersecurity Improvement Act of 2017 (<https://www.congress.gov/bill/115th-congress/senate-bill/1691/text>)) was introduced in the Senate on August 1, 2017. The bill's intent is to formalize and regulate the minimal security standards that must be met for internet-connected devices to be sold to US federal agencies. While the bill has yet to become law, the provisions clearly show the degree of regulation for IoT security that is being considered.

The bill specifically calls out the following requirements for a contractor providing federal IoT solutions:

- Solutions hardware, software, and firmware must be free of vulnerabilities as stated in the NIST National Vulnerability Database.
- Software and firmware must be capable of accepting authenticated updates and patches. Additionally, contractors must patch vulnerabilities in a timely manner. The contractor is also responsible for supporting deployments and stating when support will end and how support will be managed for IoT devices.
- Only non-deprecated protocols and technologies for communications, encryption, and interconnection should be used.
- No hardcoded credentials installed for remote administration.
- A method must be provided for any internet-connected device software or firmware to be updated or patched in any part to fix a vulnerability.
- Written certification that the IoT device provides third-party technologies must also comply with these standards.
- The Director of Management and Budget may issue deadlines for the removal and replacement of existing IoT devices that are deemed to be insecure.
- 60 days after becoming law, the National Protection and Programs Director with the assistance of private and academic technologists will issue formal cybersecurity guidelines for any and all IoT devices used by the federal government.

Written into the bill are provisions to allow for agencies to adopt or continue to use better security technologies than what is written into the bill (upon approval from the director of management and budget). Additionally, the bill recognizes some IoT devices are severely constrained in terms of processing power and memory and may not be on parity with the provisions of the bill. Again, in that case, petitions for a waiver would be managed by the Director with a plan to upgrade and replace. For these non-compliant devices, the bill states the following technologies may be approved with coordination between NIST and the director to mitigate the risk:

- Software-defined network segmentation and micro-segmentation
- Operating system-level controlled containers and microservices for isolating runtimes
- Multi-factor authentication
- Intelligent network edge solutions such as gateways that can isolate and remediate threats

While the bill is still in committee at the time of writing, it certainly shows the concern of IoT security and vulnerabilities at a federal level. The bill may morph or be killed, but the significance of IoT security has gained the attention of the US government and lawmakers.

Other governmental bodies

Other agencies in the US federal government have already issued guidelines and recommendations for a litany of technologies surrounding IoT. Most notable is the **National Institute for Standards and Technology (NIST)**, which has produced several documents and guides for the security of connected devices. They also maintain national and internationally recognized standards regarding security. Supporting material can be found at <http://csrc.nist.gov>. Several important documents relating to cryptography and FIPS standards are listed here:

- NIST Special Publication 800-121 Revision 2, *Guide to Bluetooth Security*. This specifies a set of recommended security provisions for Bluetooth Classic and Bluetooth BLE: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf>.
- NIST Special Publication 800-175A, *Guidelines for Using Cryptographic Standards in the Federal Government*: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-175A.pdf>.
- NIST FIPS standards: <https://csrc.nist.gov/publications/search?requestSeriesList=3&requestStatusList=1,&requestDisplayOption=brief&requestSortOrder=5&itemsPerPage>All>.

The Department of Homeland Security provides operational binding directives for all federal agencies in terms of national security including in the area of information technology. Recent directives include 18-01, which mandates "cyber hygiene" through email policies, key management, **Domain-based Message Authentication, Reporting, and Conformance (DMARC)**, web security using HTTPS only, and other similar actions. DHS is also involved in prescriptive guidance for Congress, other agencies, and the private sector regarding cybersecurity standards: <https://www.dhs.gov/topic/cybersecurity>.

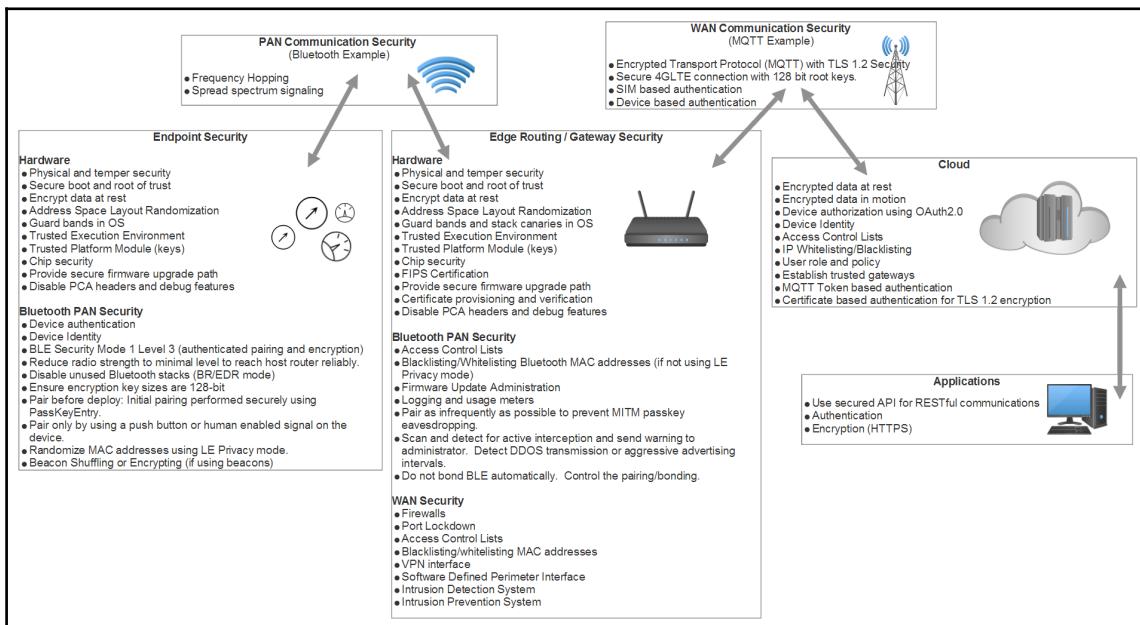
The US-CERT or Computer Emergency Response Team is also critical for anyone concerned about security. The US-CERT has been chartered with finding, isolating, informing, and stopping cybersecurity threats on a national basis since 2000. They provide digital forensics, training, real-time monitoring, reports, and actionable defenses for known zero-day exploits and active security threats. Current active alerts and mitigations can be found here: <https://www.us-cert.gov/ncas/alerts>.

IoT security best practices

Security for IoT needs to be considered from the start of design and not retrofit at the end of a program or in the field. At that point, it is too late. Security also needs to be viewed holistic from the hardware to the cloud. This section illustrates a simple IoT project from the sensor to the cloud and illustrates the "blanket" of security to consider. The intent is to deploy a system with various levels of safeguards, all there to increase the attacker's work function.

Holistic security

Narrowly focusing on one segment of IoT doesn't provide security and establishes a weak link in the security chain. One needs to establish security from the sensor to the cloud and back—a holistic approach. Each component in the chain of control and data should have a checklist of security parameters and enablers. The following diagram illustrates an example of the layers of security from the sensor to the cloud to consider in a deployment:



Sensor to cloud holistic security. Here is an example of Bluetooth-enabled sensors communicating through an edge gateway ultimately to a cloud service. Each layer needs to provide integrity and protection. Security involves both hardware and software components. This includes the security of physical devices to prevent tampering, radio signals to prevent jamming and DOS, Root of Trust hardware and ASLR to prevent malicious code injection, data by use of encryption, pairing and association by use of authentication, network through VPNs and firewalls, and so on.

Security checklist

The following is a list of security recommendations and ideas that have traditionally worked. Again, it is important to have a blanket of security:

- Use the most current operating system and libraries with all relevant patches
- Use hardware that incorporates security features like Trusted Execution Environments, Trusted Platform Modules, and non-execute spaces.
- Obfuscating code in the hopes a hacker will not reverse engineer it is relatively useless. Sign, encrypt and protect your firmware and software images, especially those freely available on a company website.
- Randomize default passwords.
- Use a Root of Trust and secure boot to ensure you have a "golden" image of software running on a customer device.
- Eliminate hardcoded passwords in ROM images.
- All IP ports must remain closed by default.
- Use Address Space Layout Randomization, Stack Canaries, and Gaurd bands in memory through modern operating systems.
- Use automatic updates. Provide manufacturers with a mechanism to fix and patch bugs and vulnerabilities in the field. This requires a modular software architecture.
- Plan for end-of-life. An IoT device may have a long usable life, but it will need to be disposed of eventually. This should include methods to securely wipe and destroy all persistent memory (flash) from the device.
- Use bug bounties programs. Reward your customers and users for finding and reporting bugs, especially defects likely to expose a zero-day exploit.
- Subscribe and participate in US-CERT active threat management to become immediately aware of active exploits and cyber threats.
- As tempting as it is to simply build a project with MQTT, HTTP, or other insecure protocols, only ship with security and authentication enabled through TLS or DTLS. Encrypt data from the sensor to the cloud.
- Employ anti-debug fuses on the package. Blow fuses in manufacturing to securely debug channels prior to releasing a product.

Summary

This chapter detailed the risks of security in the IoT. With well-known attacks such as Mirai and Stuxnet, which target IoT devices as their intended hosts, architects should design security into an IoT deployment from the start. The IoT makes for the best play space to launch an attack. The systems are usually less security mature than server and PC systems. IoT devices provide the largest attack surface on the planet. Finally, the remoteness of some IoT systems allows for attackers to be physically present and manipulate hardware, which wouldn't happen in a secure office setting. These threats need to be taken seriously, as the ramifications could have a pronounced impact on a device, a city, or a nation.

Further security links of interest are:

- Black hat: <https://www.blackhat.com>
- Defcon: <https://www.defcon.org>
- Digital Attack Map: <http://www.digitalattackmap.com>
- Gattack: <http://gattack.io>
- IDA Pro Interactive Disassembler: <https://www.hex-rays.com>
- RSA Conference: <https://www.rsaconference.comlt;/agt;>
- Shodan: <https://www.shodan.io>

We conclude the book with a list of consortiums and organizations to assist with IoT development, technologies, regulations, and standards.

13

Consortiums and Communities

Industrial consortia exist for several reasons and are essential for the promotion, governance, and construction of standards. The IoT industry is similar to other technologies and has its fair share of proprietary and open standards. This section covers the various consortia across **Personal Area Networks (PAN)**, protocols, WAN, fog and edge computing, and various umbrella consortia. A detailed and categorical description of each alliance will be detailed to help the IoT architect decide which, if any, organization is worth the time and investment to associate with. It should be noted that an organization doesn't need any involvement with an industrial alliance; many great products and businesses have been built without reliance on a consortium. However, some organizations will require corporate membership for logo usage or even the ability to productize certain standards.

A growing market segment like the IoT will spawn alliances early on in its hype cycle as a number of players vie for a stake in setting standards. This is a natural phenomenon during the rapid growth phase of any business. Oftentimes, alliances form when one similar standard competes with another, and organizations will align themselves across competitive lines. Other times, standards are defined for the industry through non-profit and academic venues. If for no other reason, the list of consortia in this chapter should help steer an architect to the resources and technical collateral needed for their design.

This chapter will provide context, history, and membership information for various organizations and consortia in the IoT space, including communications, cloud, and fog standard bodies.

PAN consortia

Personal area networks (both IP and non-IP based) have several consortia and governance committees. Many are formed by founding partners and require membership or affiliation for usage rights.

Bluetooth SIG

The details of the company are as follows:

- **Website link:** www.bluetooth.org
- **Formed:** 1998
- **Corporate membership:** 20,000

The Bluetooth SIG was formed in 1998 by five member companies: Ericsson, Nokia, Toshiba, and IBM. By the end of 1998, they had 400 members in the organization. The charter of the organization is to advance the standards, forums, markets, and understanding of Bluetooth technologies. The organization oversees the development, as well as the licensing and trademarks of Bluetooth. Organizationally, the SIG is divided into smaller focused groups: study groups for research, expert groups that span multiple Bluetooth domains, working groups that are dedicated to the development of new standards, and committees that focus on licensing and marketing. Membership is divided into promoters, who influence the overall direction and have board seats in the SIG; associate members, who can join working groups, obtain early and advanced specifications, gain marketing materials, attend PlugFest, and receive qualification listings; and adopter membership, which is the only free membership level; adopters cannot join a working group.

Thread Group

The details of the company are as follows:

- **Website link:** www.threadgroup.org
- **Formed:** 2014
- **Corporate membership:** 182

The Thread Group was originally formed by Alphabet (Google holding company), Samsung, ARM, Qualcomm, NXP, and six others. Thread is a PAN protocol based on 6LoWPAN, which is based on 802.15.4. The working group licenses Thread using the public domain BSD license model. The intent of the group was to directly counter the Zigbee protocol, especially around the use of PAN mesh networks. There are three levels of corporate membership. The bottom tier affiliate level entitles an organization to logo usage, press interviews, and access to deliverables. The mid-tier contributor level allows access to working groups and committees as well as test beds. The top-tier sponsor level provides a board seat and oversees the organizations budget.

Zigbee Alliance

The details of the company are as follows:

- **Website link:** www.zigbee.org
- **Formed:** 2002
- **Corporate membership:** 446

The Zigbee Alliance was formed around the Zigbee protocol first conceived in 1998 to address the gap in self-organizing and secure mesh networks. Zigbee sits on the 802.15.4 base layers like Thread, but is not IP-based. The software stack has remained GPL-based after several requests to provide more flexibility in licensing. There are three levels of membership. Adopters gain early access to specification, attend conferences, and can use the Zigbee logo. Participant members can propose and work in various technical committees as well as vote on specifications. Finally, promoter members receive board seats and are the sole body to ratify specifications.

Miscellaneous

Various connectivity organizations of interest include:

- **DASH7 Alliance:** Governing body of the DASH 7 protocol (www.dash7-alliance.org)
- **ModBus:** Industry consortium governing the Modbus protocol for industrial use cases (www.modbus.org)
- **Z-Wave Alliance:** Industry and governing body for Z-Wave specific technology (z-wavealliance.org)

Protocol consortia

These organizations maintain higher layer protocols and abstractions such as MQTT. While many protocols are open source, such as MQTT, membership allows for voting rights and participation in new standards.

Open Connectivity Foundation and Allseen Alliance

The details of the company are as follows:

- **Website link:** www.openconnectivity.org
- **Corporate membership:** 300
- **Founded:** 2015

The Open Connectivity Foundation was originally called the Open Interconnect Foundation but changed its name in 2016 to the OCF after the separation of Samsung from the working group and the addition of new members. For several years it was a separate entity from the Allseen Alliance, but in 2016 both organizations merged. Their combined charter is to build interoperability platforms for consumers, businesses, and industries through standards, frameworks, and a certification program under the name Open Connectivity Foundation. It spans multiple segments: automotive, consumer electronics, enterprise, healthcare, home automation, industrial, and wearables. Their frameworks are most famous for the **Universal Plug and Play (UPnP)** specification and now the IoTivity and AllJoyn connectivity framework. They use the **Internet Systems Consortium (ISC)** licensing model that is meant to be functionally equivalent to BSD. There are five levels of membership with varying degrees on yearly dues. The basic membership is free to all and grants access to test tools and read-only rights to specifications. There is a non-profit educational level that grants such organizations access to working groups and certification. Above that are the gold, platinum, and diamond levels, each with varying degrees of access from working group participation to board membership.

OASIS

The details of the company are as follows:

- **Website link:** www.oasis-open.org
- **Corporate membership:** 300
- **Founded:** 1993

OASIS is the **Organization for the Advancement of Structured Information Standards**.

OASIS is a large non-profit organization and was formed in 1993. It has been a major contributor to dozens of industry standard languages and protocols and defines the MQTT and AMQP protocols widely used in the IoT community. Their technologies touch on IoT, cloud computing, energy sectors, and emergency management to name a few. OASIS supports three types of membership. The contributor level provides unlimited committee participation. The sponsor level adds visibility and marketing benefits such as Interop demonstration and logo usage. Finally, the foundation sponsor level has the highest visibility, with perks such as OASIS presentations within a company and scholarship awards. Industry leaders representing open standards typically join at a foundational sponsor level. Company yearly dues are a function of the type of membership and the number of employees.

Object Management Group

The details of the company are as follows:

- **Website link:** www.omg.org
- **Corporate membership:** 250
- **Founded:** 1989

The **Object Management Group (OMG)** was a non-profit organization originally formed in partnership with Hewlett Packard, IBM, Sun, Apple, American Airlines, and Data General to focus on heterogeneous object standards in computing. They are best known for setting the UML standard and CORBA, and recently in the IoT arena the OMG consortium took over managing the Industrial Internet Consortium. They are involved in a number of aspects in industrial IoT, as well as **Software-Defined Networking (SDN)**. Focused IoT areas include distributed data services to ensure network interoperability as well as threat management in IoT. The group maintains a governance model with three segments: an architectural board, a platform technology committee, and a domain technology committee. There are five varying levels of membership. An analyst membership is only available to industry research firms and allows them access to a few domains and platform technology meetings. The trial membership is the least expensive option for a company and also includes invitations to a few committee meetings. The next two levels are platform and domain, which grant the holding company voting rights in their respective committees. The final membership rank is contributing member, which allows access to all committees, board seats, and so on.

IPSO Alliance

The details of the company are as follows:

- **Website link:** www.ipso-alliance.org
- **Corporate membership:** 60
- **Founded:** 2008

The group is not a standards organization but rather promotes IP enablement for *smart objects* and leads the industry on using IP to resolve interoperability issues. The group exists to complement the IETF through various working groups chartered with semantics (meta-information standards across objects), IoT protocols and the analysis of varying standards, and security and privacy.

There are three levels of membership: innovator, which is designated for an organization of fewer than 10 employees; the contributor level, which grants the company access to technical drafts and committees; and finally the promoter, which grants board seats and other perks.

Miscellaneous

Various other organizations focusing on IoT protocols and security include:

- **Online Trust Alliance:** OTA is a global non-profit developing best practices for IoT security (<https://otalliance.org>)
- **oneM2M:** Common service layer, protocols, and architecture for connected things (<http://www.onem2m.org>)

WAN consortia

These organizations cover the various long-range (LPWAN) communication and protocols. Some require membership for usage rights, while others are open protocols.

Weightless

The details of the company are as follows:

- **Website link:** www.weightless.org
- **Corporate membership:** Unknown
- **Founded:** 2012

The non-profit Weightless SIG was formed to sponsor and support the various Weightless LPWAN protocols. There are three standards supported by the SIG: Weightless-N for low-cost and long battery life, Weightless-P for two-way communication with a full set of features, and Weightless-W, which includes an extensive set of features and capabilities. The intention is to set a standard for use cases such as smart meters, vehicle tracking, and even rural broadband. Weightless focuses predominantly on usage in medical and industrial environments. While the standard is open, there is a required qualification process. Members are entitled to use Weightless IP on a royalty-free basis and have access to the certification program. Membership is fairly simple at one level: developer.

LoRa Alliance

The details of the company are as follows:

- **Website link:** www.lora-alliance.org
- **Corporate membership:** 419 (44 additional institutional members)
- **Founded:** 2014

The LoRa Alliance is a non-profit consortium that sponsors the LPWAN technology: LoRaWAN. LoRaWAN is a protocol layer architecture for long-range communication in the sub-Gigahertz spectrum. LoRaWAN typically caters to M2M and smart city deployments. There are four paid levels of corporate membership, including the Adopter level, which grants the holder rights to certified products, access to final deliverables, and invitations to certain meetings. The next level is the institutions membership, which grants the holder right to participate in working groups and gain access to early drafts. Next is the contributor level, which adds the right to vote within working groups. Finally, there is the sponsor level, which entitles the company to request a board seat, oversee operational data, and the right to chair a working group.

Internet Engineering Task Force

The details of the company are as follows:

- **Website link:** www.ietf.org
- **Corporate membership:** 1200 (general attendee counts within IETF meetings)
- **Founded:** 1986

Originally established with 21 US researchers, the IETF has grown significantly over the years, and is best known as the body controlling industry standards such as TCP/IP and various RFC documents. They now cover broad areas that span into the IoT segment, such as LPWAN protocols, 6lo, and IPVS over 802.15.4. Within the IETF exists the Internet Engineering Steering Group that governs the standardization process. It is structurally divided into seven areas, such as routing area or transport area. Each area may also host dozens of different working groups (there are over 140 working groups). Joining IETF is straightforward, and can be done by simply subscribing to and participating in a working group email list, understanding the charter and standards, and actively engaging with the broader group. The standards process is rigorous as this group defines the foundation of internet communication.

Wi-Fi Alliance

The details of the company are as follows:

- **Website link:** www.wi-fi.org
- **Corporate membership:** 700
- **Founded:** 1999

The Wi-Fi alliance is a non-profit standard body that was constructed to address the gaps of wireless interoperability in the mid-90s. With the advent of 802.11b, an industry-wide consortium was formed to build an effective governance body. The alliance controls the Wi-Fi certification process and the associated logo and trademark for devices that meet their standard. There are 19 working areas and focus groups, such as 802.11ax, security, and the IoT.

There are three levels of membership: an implementer membership allows the holder to leverage previously-certified Wi-Fi products for their end solution; affiliate membership, a unique type that affords Wi-Fi member benefits for an organization that is controlled by a sponsor or contributor member company; and contributor membership, which allows an organization to participate in certification programs and new technology definitions.

Fog and edge consortia

Fog and edge computing are becoming a growing need and a bifurcated set of industry standards are being adopted. Industrial organization and industry standards are needed to help resolve growing interoperability problems with fog computing. This section highlights some of the organizations building standards and frameworks for industry interoperability.

OpenFog

The details of the company are as follows:

- **Website link:** www.openfogconsortium.org
- **Corporate membership:** 55
- **Founded:** 2015

OpenFog is a non-profit public/private partnership that establishes standards and the adoption of fog computing. The group is chartered with influencing other standard bodies in areas such as cloud interoperability, security, and collaborating on standards. OpenFog has formed partnerships with IEEE and OPC and is governed by a 15-person board membership. There is a technical committee with six working groups under it that specialize in architecture, test beds, and other design components. They also have a marketing committee for external communication and interest generation. The organization offers a non-profit academic level of membership and a government/special interest group membership. Typical corporate memberships that can influence and participate in standards include a small company level for organizations with revenues under \$50 million.

EdgeX Foundry

The details of the company are as follows:

- **Website link:** www.edgexfoundry.org
- **Corporate membership:** 50
- **Founded:** 2017

The EdgeX Foundry is chartered with delivering an edge platform designed to resolve hardware and operating system interoperability for the IoT ecosystem through open source software. Microservices include significant middleware for rules engines, alerting, logging, registration, and device connects. The project is hosted by the Linux Foundation and is licensed under the Apache model. Dell is a principal seed for the initial code design, which is a series of microservices designed to be hardware agnostic. There are three levels of membership: associate members, silver members, and platinum members. The associate membership is limited to non-profit entities, silver members may elect one representative to the governing board, while a platinum member may have representatives on the governing board and several other committees. All members may participate in general meetings and other events.

Umbrella organizations

The following organizations govern or steer a myriad of different technical and functional aspects of IoT (as well as other segments). They represent aspects of protocols, testing, operability, technology, communication, and theory.

Industrial Internet Consortium

The details of the organization are as follows:

- **Website link:** www.iiconsortium.org
- **Corporate membership:** 258
- **Founded:** 2014

A non-profit that was started in 2014 by AT&T, Cisco, GE, IBM, and Intel, the consortium exists to bring together industry partners to assist in the adoption and development of the industrial IoT. The group is not a standards body but instead drives reference architectures and test beds for manufacturing, health, transportation, smart cities, and the energy sector. There are currently 19 working groups that span areas such as connectivity, safety, energy, smart factories, and healthcare. The group has six levels of membership including government and non-profit levels. The corporate memberships scale in influence and cost based on the company's annual sales. The testbed definitions are broad and well defined and include specific industrial use cases such as airline baggage handling tests. As mentioned previously, the OMG group manages the operations of the group but the IIC is itself its own organization.

IEEE IoT

The details of the organization are as follows:

- **Website link:** iot.ieee.org
- **Corporate membership:** N/A
- **Founded:** 2014

While not a consortium, the IEEE IoT is a special interest group under the IEEE umbrella. It is a multi-disciplinary organization comprised of academic institutions, government bodies, and industry and engineering professionals to drive IoT development. The IEEE IoT influences or hosts specific standards in the IoT world, such as 802.15 protocols and 802.11 Wi-Fi standards. The group offers free webinars, classes, and online material to help grow the knowledge base of the industry. World class conferences, workshops, and influential summits are managed by the IEEE IoT group, as well as one of the most active research journals: the IEEE Internet of Things Journal.

Miscellaneous

Various umbrella organizations of interest include:

- **Genivi:** Open software group for in-vehicle infotainment and connected cars (www.genivi.org)
- **HomeKit:** Apple consumer and mobile home automation standards (<https://developer.apple.com/homekit/>)
- **HomePlug:** Technology standards body for the connected home and consumer application (<http://www.homeplug.org/>)
- **Open Automotive Alliance:** Auto and technology group aimed at using Android in vehicles (<https://www.openautoalliance.net/#about>)
- **Wireless Life Sciences Alliance:** Connected and wireless healthcare initiative and industry (<http://wirelesslifesciences.org/>)

US government IoT and security entities

The following are government and federal organizations one should be familiar with, especially in the area of IoT security:

- **National Institute of Standards and Technology:** Defines national standards on security, encryption, and networking (<https://www.nist.gov>)
- **Department of Homeland Security:** (National Security Telecommunications Advisory Committee) is chartered with strengthening cybersecurity and global communication infrastructure (<https://www.dhs.gov/national-security-telecommunications-advisory-committee>)
- **National Telecommunications & Information Administration:** Is part of the US Department of Commerce that controls US radio spectrum allocation, domain naming, and security (<https://www.ntia.doc.gov/home>)
- **US-CERT:** Computer Emergency Response Team that identifies and responds to national high-impact computer security emergencies (<https://www.us-cert.gov/ncas/current-activity>)

Summary

Consortia and industry organizations provide significant benefits for the community in forms of standardization, technical roadmaps, and interoperability. Becoming a member allows an organization unfettered access to specifications and documentation. In many cases, membership and affiliation are necessary for usage rights. Strategically, member bodies also provide competitive strength in numbers, as various protocols and standards compete with one another in the IoT space.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Analytics for the Internet of Things (IoT)

Andrew Minteer

ISBN: 978-1-78712-073-0

- Overcome the challenges IoT data brings to analytics
- Understand the variety of transmission protocols for IoT along with their strengths and weaknesses
- Learn how data flows from the IoT device to the final data set
- Develop techniques to wring value from IoT data
- Apply geospatial analytics to IoT data
- Use machine learning as a predictive method on IoT data
- Implement best strategies to get the most from IoT analytics
- Master the economics of IoT analytics in order to optimize business value



Smart Internet of Things Projects

Agus Kurniawan

ISBN: 978-1-78646-651-8

- Implement data science in your IoT projects and build a smart temperature controller
- Create a simple machine learning application and implement decision system concepts
- Develop a vision machine using OpenCV
- Build a robot car with manual and automatic control
- Implement speech modules with your own voice commands for IoT projects
- Connect IoT to a cloud-based server

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

3

3GPP user equipment categories 236

4

4G-LTE architecture

eNodeB 243

Evolved Packet Core (EPC) 243

Evolved Universal Terrestrial Radio Access Network (E-UTRAN) 243

Public Data Network (PDN) 244

User Equipment (UE) 243

4G-LTE E-UTRAN protocol stack

about 246

access Stratum 248

Medium Access Control (MAC) 247

Non-Access Stratum 248

Packet Data Convergence Control Protocol (PDCP) 248

Physical Layer 1 247

Radio Link Control (RLC) 248

Radio Resource Control (RRC) 248

4G-LTE spectrum allocation 238

4G-LTE spectrum bands 238

5

5G

about 258

URL 259

6

6LoWPAN

about 179

edge routers 181

fragmentation 185

header compression 185

host nodes 181

mesh addressing 183

mesh-under network 183

Neighbor discovery 188

protocol stack 182

route-over network 183

router nodes 181

routing 183

security 189

topology 180

8

802.15 standards 107

A

access control list (ACL) 155, 165

Access Strata (AS) 248

acknowledged messages 144

Ad Hoc On-Demand Distance Vector (AODV) 141

Adaptive Data Rate (ADR) 265

Adaptive Frequency Hopping (AFH) 117

Adaptive Multi-Rate (AMR) 245

Additive White Gaussian Noise (AWGN) 91

Address Space Layout Randomization 436

Advanced Encryption Standard (AES) 190, 441

Advanced Message Queuing Protocol (AMQP)

332, 344

Advanced Wireless Service (AWS) 240

agricultural and environmental

about 21

use cases 21

Allseen Alliance 472

Amazon Greengrass 362, 364

Amazon Web Services (AWS) 345

Analog to Digital Converters (ADC) 205

analytics 36

application layer (APS) 161
architect 33
ARM TrustZone 434
asymmetric cryptography 445

B

Basic Rate (BR) 111
batch processing 381
Bayesian model 398, 401
Beacon Interval (BO) 150
bearers 245
Beckstrom's law 29
Binary Frequency Shift Keying (BPSK) 202
binding 333
Bit Error Rate (BER) 94, 96, 205
Bitcoin 457
Bitrate limits 89, 93
BL5 nodes, features
 friend 143
 low power 142
 proxy 142
 relay 142
blockchains
 about 456
 bitcoin 457
 Bitcoin Mining Incentives 460
 digitally signed transaction 459
 Proof of Work Burden 460
 security through peer validation 459
 security, through chaining order 461
 Unique identification 459
 use cases 457
Bluetooth 5.0 mesh addressing modes
 group addressing 144
 Unicast addressing 144
 virtual addressing 144
Bluetooth 5.0 mesh topology
 elements 142
 mesh gateway 142
 nodes 142
 unprovisioned devices 142
Bluetooth BLE 4.2
 Just Works 130
 Numeric comparison 130
 Out-of-band (OOB) 130

passkey entry 130
Bluetooth Low Energy (BLE)
 about 87, 109, 351
 random private non-resolvable 131
 random private resolvable 131
 random static 131
Bluetooth SIG
 about 470
 URL 128, 470
Bluetooth, AMP-specific stack
 AMP (PHY) 116
 AMP MAC 116
 AMP manager 116
 AMP PAL 116
Bluetooth, controller level
 Baseband resource manager 115
 BR/EDR PHY (controller block) 115
 device manager 115
 Host Controller Interface (HCI) 115
 LE PHY 115
 link controller 115
 link manager 115
Bluetooth, core architectural blocks
 AMP-specific stack 116
 controller level 115
 host level 115
Bluetooth, host level
 Attribute protocol (ATT) 116
 audio 116
 channel manager 115
 Generic Access Profile (GAP) 116
 Generic Attribute Profile (GATT) 116
 L2CAP 115
 resource manager 116
 RFCOMM 116
 Security Manager Protocol (SMP) 116
 Service Discovery Protocol (SDP) 116
Bluetooth
 about 109
 Alternative MAC/PHY (AMP) 114
 Basic Rate/Enhanced Data Rate mode
 (BR/EDR) 114
 beaconing 131
 BLE operation 123
 BLE security 130

Bluetooth 5 communication process 111
Bluetooth 5 communication topologies 111
Bluetooth 5 PHY 117
Bluetooth 5 range 137
Bluetooth 5 stack 113
Bluetooth 5.0 mesh addressing modes 144
Bluetooth 5.0 mesh provisioning 146
Bluetooth 5.0 mesh topology 141
Bluetooth mesh 139
BR/EDR operation 121
BR/EDR security 129
core architectural blocks 115
history 109
interference 117
Low Energy (LE) mode 114
packet structure 119
profiles 127
speed enhancement 137
Border Gateway Protocol (BGP) 282
Business Rule Management Systems (BRMS) 378

C

Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) 149
CBC, with Message Authentication Code (CBC-MAC) 442
ceilometer 350
cellular connectivity
 3GPP user equipment categories 236
 4G-LTE architecture 242
 4G-LTE E-UTRAN protocol stack 246
 4G-LTE geographical areas 248
 4G-LTE packet structure 251
 4G-LTE spectrum allocation 238
 4G-LTE spectrum bands 238
 4G-LTE topology 242
5G 258
about 230
cellular access technologies 235
dataflow 248
governance models 231
handover procedures 248
LTE Cat-0 252
LTE Cat-1 253
LTE Cat-M1 (eMTC) 254

LTE Cat-NB 256
standards 231
Cellular Internet of Things (CIoT) 256
Certificate Authorities (CA) 450
Chain Reaction 431
Channel Selection Algorithm 2 (CSA2) 119
charge-coupled devices (CCD) 56
Chirp Spread Spectrum (CSS) 265
Cinder 349
Cipher Block Chaining (CBC) 442
Cipher Feedback Chaining (CFB) 442
Classification and Regression Test (CART) 396
classification
 about 393
 binomial 393
 multi-class 393
clear channel assessment (CCA) 151
cloud architectures
 constraints 351
 latency effect 352
cloud services model
 about 339
 IaaS 341
 NaaS 340
 PaaS 341
 SaaS 341
CloudPing
 URL 353
CoAP, actors
 client 324
 endpoints 324
 intermediary 324
 observers 324
 origin servers 324
 proxies 324
 server 324
CoAP, messaging formats
 acknowledgement (ACK) 326
 confirmable (CON) 326
 non-confirmable (NON) 326
 Reset (RST) 326
Code Division Multiple Access (CDMA) 236
coherence bandwidth 99
cold junction compensation 41
communication theory

about 82
RF energy 82, 86
RF interference 87
theoretical range 82, 86
complementary metal-oxide (CMOS) devices 56
complex event processing (CEP) 384
Constrained Application Protocol (CoAP)
 about 192, 321
 actors 324
 architecture 322
 messaging formats 325, 328, 329
 Request/Response layer 322
 Transactional layer 323
 usage example 330
Constrained RESTful Environments (CoRE) 321
consumer-based devices
 about 17
 use cases 18
contention access periods (CAP) 150
contention free periods (CFP) 151
control plane 246
Convolutional Neural Networks (CNN)
 about 401
 backpropagation 407
 example 405
 filters 402
 formal description, on forwarding propagation
 403
 hidden layers 403
 layer 402
 max pooling 402
 subsampling 402
 training 407
Correlation Power Analysis (CPA) 432
Counter (CTR) 442
cryptocurrency
 about 456
 IOTA 462
cryptography
 about 439
 asymmetric cryptography 445
 hash 449
 Public key infrastructure (PKI) 450
 Symmetric cryptography 441
 transport layer security (TLS) 452

current sensors 44
cybersecurity, defense mechanisms
 Address Space Layout Randomization 425
 Blackhole (sinkhole) 425
 Data Execution Prevention 426
 Deep Packet Inspection (DPI) 426
 firewall 426
 guard bands 426
 honeypots 426
 Instruction-Based Memory Access Control 426
 Intrusion Detection System 426
 Intrusion Prevention System 426
 milkers 426
 non-executable memory 426
 port scanning 426
 private key 427
 public key 426
 Public Key Infrastructure (PKI) 426
 root of trust 427
 secure boot 427
 stack canaries 427
 trusted execution environment 427
cybersecurity
 about 423
 amplification attack 423
 ARP spoof 423
 attack 423
 banner scans 423
 botnets 423
 Brute Force 423
 Buffer Overflow 423
 C2 423
 Correlation Power Analysis Attack 424
 defense 425
 dictionary attack 424
 Distributed Denial of Service (DDOS) 424
 fuzzing 424
 Man-in-the-Middle Attack (MITM) 424
 NOP sleds 424
 RCE Exploit 424
 Replay Attack (also known as a Playback Attack)
 424
 Return-Oriented Programming (ROP Attack)
 424
 Return-to-libc 425

rootkit 425
side channel attack 425
spoofing 425
SYN Flood 425
threats 423
Zero-Day Exploits 425

D

DASH7 Alliance
 URL 471
data analytics
 about 373
 alerting 373
 batch queries 375
 comparing, with machine learning 419
 complex event processing (CEP) 376, 384
 control 376
Convolutional Neural Networks (CNN) 401
databases 374
deep analytics pathway 375
errors 374
ingestion 381, 384
joins 374
Lambda architecture 385
lambda architecture 377
models 375
patterns 374
preprocessing 373
rules engines 376, 378, 381
sector, use cases 386
signaling 375
stream processing 376
temporal events 374
top-level cloud pipeline 376
tracking 374
training 375
trends 375
windowing 374
data communication 35
Datagram Transport Layer Security (DTLS) 190, 453
decibel (dB) 83
decision tree 396
Dedicated Short Range Communication (DSRC) 218

Defense Information Systems Agency (DISA) 454
demilitarized zone (DMZ) 293
Department of Homeland Security
 URL 480
Device to Device (D2D) communication 263
Differential Phase Shift Keying (DPSK) 203
Differential Power Analysis (DPA) 432
Differential Quaternary Phase Shift Keying (DQPSK) 117
Differentiated Services (DiffServ) 291
digital light processor (DLP) 50
Digital to Analog Converters (DAC) 205
direct exchange 333
Direct Sequence Spread Spectrum (DSSS) 151, 201
Direct Tunnel Setup (DTS) 250
Distribution System (DS) 199
Domain-based Message Authentication 465
Duplicate Address Detection (DAD) 188

E

E-UTRAN Cell Identity (ECI) 249
E-UTRAN Cell Global Identifier (ECGI) 249
Eddystone
 Eddystone-EID 133
 Eddystone-TLM 133
 Eddystone-UID 133
 Eddystone-URL 133
edge compute 36
Edge computing
 versus cloud computing 355
Edge consortia
 about 477
 EdgeX Foundry 477
EdgeX Foundry
 about 478
 URL 478
Electronic Codebook (ECB) 442
Elliptic Curve Cryptography (ECC) 447
elliptical-curve Diffie-Hellman (ECDH) 130
encryption
 Cryptographic hash 439
 Public Key encryption 439
 Symmetric Key encryption 439
endpoints 39

energy harvesting
about 67
piezo-mechanical harvesting 70
RF energy harvesting 71
solar harvesting 69
thermal harvesting 72
energy sources 65
energy storage
about 73
batteries 77
energy model 73
power model 73
power sources 78
radioactive power sources 78
summary 78
supercapacitors 77
energy
about 22
use cases 22
Enhanced Data Rate Mode (EDR) 110
EPS Bearer 245
error carousel 414
Everything as a Service (XaaS) 339
Evolved Packet Core (EPC)
Home Subscriber Server (HSS) 243
Mobility Management Equipment (MME) 243
Policy Control and Charging Rules Function (PCRF) 244
Public Data Network Gateway (PGW) 244
Servicing Gateway (SGW) 243
Evolved Packet System (EPS) 256
Extended Discontinuous Receive (eDRX) mode 255
Extended Inquiry Response (EIR) 110

F

fading phenomenon
fast fading 98
slow fading 98
Fast Fourier Transform (FFT) 205, 416
Federal Communications Commission (FCC) 102
Field Area Network (FAN) 81
Field Effect Transistor (FET) 46
Field-programmable gate array (FPGAs) 418
filter

content filtering 306
subject filtering 306
type filtering 306
financial 18
fog compute 36
Fog computing, factors
data volume reduction 365
fog node capabilities 365
number of edge devices 365
system reliability 366
Fog computing
about 354
Amazon Greengrass 362, 364
factors 365
Hadoop philosophy 354
Lambda 362, 364
OpenFog Reference Architecture 356
topologies 365, 370
versus cloud computing 355
versus Edge computing 355
Fog consortia
about 477
OpenFog 477
forward error correction (FEC) 94
Frame Check Sequence (FCS) 274
Free Space Path Loss (FSPL) 84
Frequency Division Duplex (FDD) 238
Frequency Division Multiple Access (FDMA) 118, 235
Frequency hopping spread spectrum (AFH) 110
frequency-hopping spread spectrum (FHSS) 117
full end devices (FEDs) 192
functional block, responsibilities
analog-to-digital conversion 57
chroma subsampling 58
color space conversion 58
color space conversion 3 x 3 58
dead pixel correction 57
debayer filtering 58
demosaicing 58
gamma correction 58
JPEG encoder 58
noise reduction 58
optical clamp 57
sharpening 58

white balance 57

G

gateway functions 280

GATT

characteristics 128

services 127

Gaussian Frequency Shift Keying (GFSK) 117, 202, 272

General-Purpose Computation on Graphics

Hardware (GPGPU) 418

Genivi

URL 479

glance 345

Global System for Mobile Communications (GSM) 233

Google Cloud Platform (GCP) 315

Google IoT Core

URL 315

government and military

about 24

use cases 24

GPRS Tunneling Protocol (GTP) 245

gradient 409

Greengrass 362

Guard Interval (GI) 205

H

Hall voltage 44

hardware security

about 433

key management 435

memory space 436

physical security 437

processor 436

Root of Trust 433

storage security 436

trusted platform modules 435

hashing functions 449

healthcare

about 19

use cases 19

heat 350

Home Area Network (HAN) 81

HomeKit

URL 479

HomePlug

URL 479

Horizon 349

Host Controller Interface (HCI) 113

Host Identity Protocol (HIP) 290

hybrid cloud 343

hyper-frames (HF) 255

hyperplanes

URL 393

I

iBeacons

monitoring 132

ranging 132

URL 135

Idaho 335

Identity Resolving Key (IRK) 131

IEEE 802.11, authentication types

WEP 215

WPA 215

WPA-PSK 215

WPA2 216

IEEE 802.11, encoding techniques

Amplitude Shift Keying (ASK) 202

Frequency Shift Keying (FSK) 202

Phase Shift Keying (PSK) 203

IEEE 802.11, interference mitigation techniques

Direct sequence spread spectrum 205

Frequency Hopping Spread Spectrum (FHSS) 205

Orthogonal Frequency Divisional Multiplexing (OFDM) 205

IEEE 802.11

ad hoc 199

architecture 198

distribution system 199

encoding techniques 202

Explicit feedback beamforming 209

IEEE 802.11ac 216

IEEE 802.11ah 222

Implicit feedback beamforming 209

infrastructure 199

MIMO 207

modulation 202
operation 213
packet structure 210
protocols 196
security 215
spectrum allocation 200
suite of protocols 197
vehicle-to-vehicle 218

IEEE 802.11ah, power saving states
Non-TIM stations 226
Traffic Indication Map (TIM) 226
Unscheduled stations 226

IEEE 802.11ah, stations
relay node 225
root access point 225
STA 225

IEEE 802.15.4
about 147
address modes 153
architecture 147
Full function device (FFD) 152
packet structure 153
Reduced function device (RFD) 152
security 155
start-up sequence 154
topology 152

IEEE IoT
about 479
URL 479

if this, then that (IFTTT) protocol 7

image signal processor (ISP) 57

In-Band (IB) 358

Independent Basic Set Service (IBSS) 199

industrial and manufacturing
about 16
use cases 17

Industrial Internet Consortium
about 478
URL 478

Industrial IoT (IIoT) 16

Industrial, Scientific and Medical (ISM) devices 105

Information Technology (IT) 179

information theory
about 88

Bit error rate (BER) 94, 96
bitrate limits 89, 93
narrowband communication, versus wideband communication 97
Shannon-Hartley theorem 89, 93

Infrastructure as a Service (IaaS) 339, 341

Infrastructure Basic Set Service (BSS) 199

ingestion 381, 384

Initialization Vector (IV) 442, 449

input devices 60

Intel Boot Guard 435

Inter-LTE Handover 249

Inter-symbol Interference (ISI) 205

International Mobile Telecommunications-2000 (IMT-2000) 231

International Mobile Telecommunications-Advanced (IMT-Advanced) 231

International Telecommunication Union (ITU) 101, 231

Internet Engineering Task Force
about 476
URL 476

Internet Key Exchange (IKE) 289

Internet of Things (IoT)
about 7, 39, 422
agricultural and environmental 21
agricultural and environmental, use cases 21
consumer 17
consumer, use cases 18
energy 22
energy, use cases 22
financial 18
government and military 24
government and military, use cases 24
health care, use cases 19
healthcare 19
history 11
industrial and manufacturing 16
industrial and manufacturing, impact 17
industrial and manufacturing, use cases 17
marketing 18
potential 13
references 12
retail 18
retail, use cases 18

smart city 22
smart city, use cases 23
transportation and logistics 20
transportation and logistics, use cases 20
versus machine or machine 28

Internet protocol (IP)
about 176
manageability 177
reliability 177
scalability 176
standards-based 176
ubiquity 176
using, in IoT 176

internet routing 36

Internet Systems Consortium (ISC) 472

Intra-LTE Handover 249

IoT architecture 32

IoT cyber attacks
anatomy 427
attack spread 430
Chain Reaction 431
destruction 430
Mirai 428
Stuxnet 430

IoT endpoints
about 55
vision system 56

IoT security, best practices
about 466
checklist 467
holistic security 466

IoT, roles
aggregators 27
cloud 27
data analytics 27
gateways 27
local area networks 27
routers 27
security 27
sensor communication systems 27
sensors 27
WAN 27

IOTA
about 462
URL 463

IPSO Alliance
about 474
URL 474

J

JSON Web Tokens (JWT) 315
Just Works 129

K

key derivation function (KDF) 439
Key Management Protocol (KMP) 108
keystone 345

L

Lambda 364
Lambda architecture
batch layer 385
service layer 386
speed layer 386

Last Will and Testament (LWT) 309

latency effect 352

Licensed-Assisted Access (LAA) 241

light-out management (LOM) 285

linear classifier 394

Linear Time Invariant (LTI) 270

listen-before-talk (LBT) 241

Long Range Radio (LoRa) 229

Long Short-Term Memory (LSTM) 414

Long-Term Evolution (LTE) 231

long-term key (LTK) 130

look-up table (LUT) 58

LoRa Alliance
about 475
URL 475

LoRa
about 263
physical layer 264

LoRaWAN
about 263
MAC layer 266
summary 269
topology 268

Low Energy (LE) 110, 111

low power node (LPN) 142

Low Power Wide Area Networks (LPWAN) 252
LTE Unlicensed (LTE-U) 241

LTE
cyclic prefix 238
resource block 238
resource element 238
Subcarrier spacing 238
time slot 238

M

machine learning
about 36, 387, 391
assessment 419
Bayesian model 401
Bayesian models 398
classification 393
comparing, with data analytics 419
inference 418
models 392
random forest 396, 398
regression 396
RNN 411, 418
training 418
machine-to-machine (M2M) communication 28,
 112, 321
Man-in-the-Middle attacks (MITM) 129
marketing 18
Massive-MIMO (M-MIMO) 261
master (M) 112
Maximum Coupling Loss (MCL) 96, 257
Maximum Transmission Unit (MTU) 185
Mean Opinion Score (MOS) 291
Media Access Controller (MAC) 196
Medium Access Controller (MAC) 191
Mesh link establishment (MLE) 193
mesh-specific layers
 access layer 140
 bearer layer 140
 BLE 140
 foundation models 139
 lower transport layer 140
 Models 139
 network layer 140
 upper transport layer 140
message authentication codes (MAC) 439

Message Orientated Middleware (MOM) 303
Message Queue Telemetry Transport (MQTT)

about 302, 305
architecture 308
communication formats 312, 313, 314
packet structure 311
publish-subscribe 306
quality of service 310
URL 305
versus MQTT-SN 321
working 315, 317

Metcalfe's law 29

Micro-electromechanical systems (MEMS)

about 27, 50
accelerometers 50
gyroscopes 50
microphones 53
pressure sensors 54
millimeter waves (mmWave) 260

MIMO

about 207
spatial diversity 207
spatial multiplexing 207

minimal end devices (MEDs) 192

Mirai

about 428
Brute Force Telnet 428
infect 428
victims, scanning 428

Mobile Terminations (MT) 243

Mobile wireless service (MWS) 110

ModBus

URL 471

Modulation and Coding Scheme (MCS) 206

more data (MD) 143

MQTT, quality of service

 QoS-0 (non-assured transmission) 310
 QoS-1 (assured transmission) 310
 QoS-2 (assured service on applications) 310

MQTT-SN, topology

 brokers 319
 clients 318
 forwarders 318
 gateways 318

MQTT-SN

about 318
aggregating gateways 319
architecture 318
gateway advertisement 320
gateway discovery 320
transparent gateway 319
versus MQTT 321
Multicast Protocol for Low Power and Lossy Networks (MPL) 193
multiple input, multiple output (MIMO) 90

N

narrowband communication
versus wideband communication 97, 101
National Institute of Standards and Technology
URL 480
National Telecommunications and Information Administration (NTIA)
about 102
URL 480
Neighbor Confirmation (NC) 188
Neighbor discovery (ND) 188, 195
Neighbor Registration (NR) 188
Neighborhood Area Network (NAN) 81
Network Discovery (ND) 181
Network Function Virtualization (NFV) 296
network layer (NWK) 161
Networking as a Service (NaaS) 340
Neutron 348
NIST Special Publication
URL 465
nodes 458
Non-Access Stratum (NAS) 243, 245, 248
Non-Execution or Executable Space Protection 436
non-standalone (NSA) 259
Nova compute 346, 348
numeric comparison 129

O

Object Management Group (OMG)
about 473
URL 473
on-board unit (OBU) 218

oneM2M
URL 474
Online Trust Alliance
URL 474
Open Automotive Alliance
URL 479
Open Connectivity Foundation 472
Open Shortest Path First (OSPF) 282
OpenFog Reference Architecture
about 356
accelerators 360
actuators 362
application services 357
application support 358
compute 361
control systems 362
hardware platform infrastructure 361
hardware virtualization 359
network 360
node management 358
node security 359
protocol abstraction 362
sensors 362
software backplane 358
storage 361
URL 356
OpenFog
about 477
URL 477
OpenStack cloud architecture
about 343, 344
ceilometer 350
Cinder 349
glance 345
heat 350
Horizon 349
keystone 345
Neutron 348
Nova compute 346, 348
Swift 348
Operational Technology (OT) 179
Organization for the Advancement of Structured Information Standards (OASIS)
about 473
URL 473

Organization Unique Identifier (OUI) 119
Orthogonal Sequence Spread Spectrum (OSSS) 272
Orthogonal Frequency Divisional Multiplexing (OFDM) 201
Out-of-band (OOB) 129
Out-of-Band Management (OOBM) 285
output devices 60
Output Feedback Chaining (OFB) 442
Over-the-Air-Activation (OTAA) 267

P

PAN consortia
about 469
Bluetooth SIG 470
Thread Group 470
Zigbee Alliance 471
parked (P) 112
passkey entry 129
Peer Aware Communications (PAC) 108
perceptrons 387
Performance Enhancing Proxies (PEP) 291
Periodic Restricted Access Windows (PRAW) 226
Personal Area Networks (PAN) 35, 106, 469
Personal Identification Number (PIN) 129
photoelectric sensors 46
physical security 437
piconet 112
piezo-mechanical harvesting 70
Platform as a Service (PaaS) 339, 341
power 34
power efficiency 91
Power Limited region 93
power management 66
Power Save Mode (PSM) 253
private cloud 342
Protocol consortia
about 471
Allseen Alliance 472
IPSO Alliance 474
OASIS 472
Object Management Group (OMG) 473
Open Connectivity Foundation 472
protocol
about 36, 302

comparison 336
summary 336
pseudo-random number generator (PRNG) 447
public cloud 343
Public Key Infrastructure (PKI) 450
Public Land Mobile Network (PLMN) 244
publisher 306
pulse density modulated (PDM) 54
Pyroelectric Infrared (PIR) sensors 46

Q

Quadrature Amplitude Modulation (QAM) 204
Quality of Service (QoS) 244, 290

R

Radio Access Networks (RANs) 242
Radio Resource Control (RRC) 246
radio spectrum
about 101
governing structure 101, 105
radioactive power sources 78
Radiocommunication Sector (ITU-R) 231
random forest 396, 398
Random Frequency and Time Division Multiple Access (RFTDMA) 272
Received Signal Strength Indicator (RSSI) 110
Registration Authority (RA) 450
regression 396
remote procedure call (RPC) 344
Reporting 465
Resistance Temperature Detectors (RTD) 42
Restricted Access Window (RAW) 224
retail
about 18
use cases 18
RF energy harvesting 71
RIPng 283
RNN 411, 418
road-side unit (RSU) 218
Root of Trust (RoT) 433
round-trip times (RTT) 353
Router Advertisement (RA) 188
Router Solicitation (RS) 188
Router-eligible End Devices (REED) 192

routing functions
 about 280
 analytics 294
 edge processing 295
 failover 285
 gateway functions 280
 metrics 294
 Out-of-Band Management 285
 QoS 290
 routing 280
 security functions 293
 traffic shaping 290
 VLAN 286
 VPN 288
routing
 about 280
 broadcast routing 282
 distance vector routing 281
 flooding 281
 flow-based routing 281
 hierarchical routing 281
 link state routing 281
 multicast routing 282
 shortest path routing 281
rules engines 378

S

scan mechanisms
 active scanning 214
 passive scanning 213
scatternet 112
sector
 use cases 387
Secure Simple Pairing (SSP) 110
security 37
Seebeck electromotive effect 40
segmentation and reassembly (SAR) 140
semi-supervised learning 392
sensing 34
sensing devices
 about 40
 current sensors 44
 hall effect sensors 44
 LiDAR 48
 MEMS sensors 50

photoelectric sensors 46
PIR sensors 46
temperature sensing 40
thermocouples 40
sensor fusion
 about 59
 centralized 59
 de-centralized 59
sensors
 controller, using 64
 implementing 61
 TI sensorTag CC2650 61
Service Access Points (SAP) 150
Service Set Identifier (SSID) 195
Session Initiation Protocol (SIP) 245
set 396
Shannon limit 91
Shannon-Hartley theorem 89, 93
Siddhi 385
Sigfox
 about 270
 MAC layer 273
 physical layer 271
 protocol stack 274
 topology 275
signal-to-noise ratio (SNR) 90
Simple Text Message-Oriented Middleware
 Protocol (STOMP) 332
slave 112
Slot Availability Masks (SAM) 110, 119
smart city
 about 22
 use cases 23
Software as a Service (SaaS) 339, 341
software security 433
Software-Defined Networking (SDN)
 about 290, 296, 340, 473
 architecture 297
 bandwidth calendaring 300
 benefits 300
 dynamic load management 300
 internetworking 299
 service chaining 300
Software-Defined Perimeter (SDP)
 about 300, 454

architecture 454
solar harvesting 69
Spatial Diversity Multiple Access (SDMA) 216
Spatial Division Multiplexing (SDM) 216
Spectrum Resource Utilization (SRU) 108
standalone (SA) 259
standby (SB) 112
stateful 378
Static point of Interest (POI) 131
storage security 436
Stuxnet
 about 430
 initial infection 430
 URL 431
 windows attack 430
subscriber 306
supercapacitors
 about 77
 electric double layer capacitors (EDLC) 77
 psuedocapacitors 77
SuperFrame Interval (SO) 150
supervised learning 392
swift 348
Symmetric cryptography 441
System Architecture Evolution (SEA) 242

T

Target Beacon Transmit Time (TBTT) 214
Target Wake Time (TWT) 226
Technical Specification Groups (TSG) 233
temperature sensing 40
Temporal Key Integrity Protocol (TKIP) 215
TensorFlow
 URL 405
thermal harvesting
 thermionic 72
 thermoelectric 72
thermistor 43
thermocouple (TC)
 about 40
 Resistance Temperature Detectors (RTD) 42
 temperature sensor summary 44
 thermistors 43
thermoelectric generator (TEG) 72
Thread architecture

border router 192
end devices 192
REEDs 192
sleepy end devices 192
thread routers 192
Thread Group 470
Thread
 about 190
 addressing 194
 architecture 191
 Neighbor discovery (ND) 195
 protocol stack 192
 routing 193
 topology 191
threat 37
TI sensorTag CC2650 61
time diversity 98
Time Division Duplex (TDD) 117, 238
Time Division Multiple Access (TDMA) 118, 236
Time of Flight (TOF) 49
time-to-live field (TTL) 143
Tracking Area (TA) 243
Tracking Area Identity (TAI) 249
Tracking Area Update (TAU) 253
transport blocks (TB) 247
Transport Layer Security (TLS) 190, 452
transportation and logistics
 about 20
 use cases 20
Trusted Platform Module (TPM) 435

U

Ultra Narrow Band (UNB) 271
Umbrella organizations
 about 478
 IEEE IoT 479
 Industrial Internet Consortium 478
unacknowledged messages 144
Universal Integrated Circuit Card (UICC) 243
Universal Plug and Play (UPnP) 472
Universal Resource Identifier (URI) 303
Universal Resource Locator (URL) 303
unsupervised learning 392
US Congressional Bill
 about 463

URL 463
US-CERT
 URL 480
user equipment (UE) 238

V

Validation Authority (VA) 451
value of a network 29
vehicle-to-infrastructure (V2I) model 218
vehicle-to-vehicle (V2V) 218
VLAN 286
Voice over Internet Protocol (VOIP) 291
Voice over IP (VOIP) 245
VPN
 about 288
 Generic Routing Encapsulation (GRE) 288
 Internet Protocol Security (IPSec) VPN 288
 Layer 2 Tunneling Protocol (L2TP) 289

W

WAN consortia
 about 474
Internet Engineering Task Force 476
LoRa Alliance 475
weightless 475
Wi-Fi Alliance 476
weather 335
Weightless
 about 475
 URL 475
Wi-Fi Alliance
 about 476
 URL 476
Wide Area Network (WAN) 229
wideband communication
 versus narrowband communication 97, 101
Wireless Access in Vehicular Environments (WAVE) 220
Wireless Body Area Network (WBAN) 81
Wireless Life Sciences Alliance
 URL 479
Wireless Local Area Network (WLAN) 81, 229
Wireless Mobility Internet Access (WiMAX) 245
Wireless Personal Area Networks (WPANs)

802.15 standards 107
about 106, 229
Bluetooth 109
IEEE 802.15.4 147
standards 107
Z-Wave 167
Zigbee 157
WLAN 196
Working Groups (WG) 233

Z

Z-Wave Alliance
 URL 471
Z-Wave, controllers
 bridge controller 169
 installer controller 169
 status update controller (SUC) 169
 SUC ID server (SIS) 169
Z-Wave, slaves
 enhanced slave 169
 routing slave 169
Z-Wave
 about 167
 addressing 171
 application layer 170
 controller device 168
 MAC layer 170
 overview 167
 PHY layer 170
 protocol stack 170
 routing 172
 routing layer 170
 slave device/node 168
 topology 172
 transfer layer 170
Zigbee Alliance
 about 471
 URL 471
Zigbee coordinator (ZC) 165
Zigbee device object (ZDO) 161
Zigbee End Devices (ZED) 159, 165
Zigbee
 about 157
 addressing 162
 association 165

broadcasting 164
cluster tree 159
history 157
link key 166
long address (64 bit) 163
MAC 160
master key 166
mesh network 159
mesh routing 164
mesh routing (table routing) 164
network key 166

overview 158
packet structure 162
protocol stack 161
security 165
short address (16 bit) 163
source routing 164
star network 159
tree routing 164
Zigbee controller (ZC) 158
Zigbee End Devices (ZED) 158
Zigbee PHY 160
Zigbee router (ZR) 158