Volume 2, Issue 3

Methods

Mouser's technology & solutions eZine

THE SMARTER EDGE

In This Issue

Methods
Mouser's technology & solutions eZine

Executive Editor
Deborah S. Ray

Contributing Authors
Charles Byers
Steven Evanczuk
Jeff Fellinge
Paul Golata
Bin Jiang
M. Tim Jones
Chaofan Ma
Jason Shepherd
Houbing Song
Huifang Xu

Technical Contributor
Paul Golata

Editiorial Contributor
LaKayla Garrett

Design & Production
Robert Harper

With Special Thanks
Kevin Hess
Sr. VP, Marketing

Russell Rasor
VP, Supplier Marketing

Jack Johnston, Director
Marketing Communication

Raymond Yin, Director
Technical Content

Foreword: Realizing the Holy Grail of Digital via a Smarter Edge

by Jason Shepherd, CTO of IoT and Edge Computing for Dell Technologies

In the history of computing, the pendulum has faithfully swung every 10 to 15 years between centralized and distributed models. However, given the sheer volume of networked devices going forward, we need a smarter edge because it simply isn't feasible to send all data directly to the cloud.

IoT Devices Can Be Like Teens That Don't Pay the Mounting Phone Bill

The articles within this issue of *Methods* talk about latency, bandwidth, and security as key technical reasons for a smarter edge. However, there's a big kicker—the "total lifecycle cost" of data. People that start with cloud-centric analytics often quickly realize that it gets expensive fast when chatty IoT devices hit public-cloud application-programming interfaces (APIs).

The majority of IoT data is "perishable," meaning if you don't create an alert or take action based on the data in the moment, it's not going to do you much good later. A smarter edge will enable a combination of taking action, storing, forwarding, and scrapping data on the spot.

There are Many Edges

So what's the "edge?" The fact is, when you read the articles in this issue, you'll see that there isn't one edge; rather there's a collection of distributed edges.

My simple definition is that edge computing is about locating computing resources that are necessary and feasible to the subscribers (that is, the users and devices) needing it. Here are some examples:

The closest edge to subscribers in which a telecommunications company can feasibly locate significant computing resources at the bottom of their cell towers or at their baseband units. The article titled "Could the Edge End the Connectivity Wars?" talks about this hot topic.

On-prem edges, which include traditional and micromodular data centers, hyper-converged infrastructure and edge gateways, hubs, and routers, are utilized when it's necessary to be on the same local network as field devices. These edges reduce latency and bandwidth consumption and maximize security and uptime for mission-critical applications.

The device edge includes sensors, actuators, and controllers in the field, which gather data from the physical world and run processes. The article titled "A View from the Edge" talks extensively about microcontroller unit (MCU)-based smart sensors that populate the device edge.

The Fog vs. the Cloud

The term "fog" is foggy in the minds of many people. Simply put, the fog is the combination of all the edges and the networks in between effectively everything but the cloud. The article titled "Edge Design and Deployment: Intermediate Levels of Computation, Networking, and Storage" discusses fog nodes that span a variety of edges, working together as a larger computing continuum in real time.

Regardless of how we label things, we need scalable solutions for a smarter, distributed edge that works together with public, private, and hybrid clouds while meeting the needs of operational technology (OT) and information technology (IT) organizations alike.

We Need a Cloud-Native Edge for the IoT Scale

Key to the concept of cloud-native is the utilization of modern development and operations (DevOps), continuous delivery of data, loosely-coupled microservices, and overall platform-independence.

However, cloud-native is more about how software is built and deployed than where it's actually run.

It's only logical that the same reasons that these principles help companies develop and deploy massively scalable applications in the cloud also make the same principles highly applicable across all the different edges. In fact, I would contend that these principles are necessary to enable smarter edge designs and help enterprises continuously define software better as well as create more innovative outcomes spanning from edge to cloud, all to stay competitive in an ever-changing market.

Facilitating an Open, Cloud-Native Edge Ecosystem

Launched last year by the Linux Foundation, the vendor-neutral, open-source EdgeX Foundry project is building an open framework for edge computing to facilitate an interoperable cloud-native IoT edge. EdgeX brings together devices, speaking any mix of connectivity protocols, with a growing ecosystem of value-added applications in an inherently multi-edge and multi-cloud world.

Another important effort that will help us reach a smarter edge is the recently-launched Akraino project, which is focused on facilitating open interoperability at the telco edge infrastructure layer and which will eventually extend down to a device edge. EdgeX and Akraino are highly complementary efforts and are spinning up collaboration to make sure related APIs are correlated.

Realizing the Holy Grail of Digital

I'll close with what I see as the "Holy Grail of Digital"—selling data, resources (for computing, networking, energy, etc.) and services (such as domain-specific consulting) to people you don't even know.

Over time, by combining the silicon-based root of trust; connectivity standards; open, defacto-standard APIs, established by projects like EdgeX and Akraino; and ledger technologies (like blockchain) in a smarter edge, we'll build the intrinsic, pervasive trust needed for this soon coming digital phenomenon. This will enable us to create data in the physical world, send it out into the ether, and then sit back and collect checks from complete strangers on our terms. This is the scale factor!

We'll never realize this new phenomenon with the current plethora of proprietary IoT platforms that are trying to lock customers in, thinking they can then sell their data. Yes, we'll see walled gardens in consumer products, but these walls simply do not work for any single entity to own the trust of businesses and consumers, especially when it comes to scaling a system of systems and supporting business-to-business (B2B) and business-to-business-to-consumer (B2B2C) use cases that span private and public domains.

My advice is to start small, while also investing in open technologies that facilitate scale and pervasive trust. We live in exciting times, and I can't wait to see how the world makes use of a smarter edge! M

A View from the Edge: An Introduction to the Smarter Edge

by Stephen Evanczuk for Mouser Electronics

A new view of IoT edge computing finds these systems serving an expanded role well beyond that of data concentrators, network routers, or connection hubs used in smart homes. As both IoT devices and cloud resources become more sophisticated, this emerging view elevates edge devices to stand as a critical component needed to manage complexity, enhance performance, and meet evolving end-to-end requirements of IoT applications.

In the Internet of Things (IoT), edge computing devices fill a role that on the first impression parallels familiar network appliances such as Wi-Fi routers, smart home hubs, data concentrators, and more. The varied demands of diverse IoT applications drive a more expansive view of edge computing as it evolves to play a central role in end-to-end IoT applications and their underlying platforms. Increasingly, IoT edge devices are an integral part of a fluid application platform that binds vast numbers of peripheral IoT devices with local- and cloud-based resources into a complete IoT application (Figure 1).

With expectations of billions of sensor devices closely monitoring every detail of life and work, the IoT presents unparalleled opportunities for building a deep understanding of the effect of changes in the environment, in industrial processes, individual products, and ourselves. The simultaneous rise of both advanced IoT devices and cloud-based computing resources offers an extraordinary platform for developing applications able to deliver this understanding. The very sophistication of those devices and resources can paradoxically overwhelm efforts to realize the full potential of IoT applications.

Enhanced Connectivity

On the device side, increased integration of sensors and actuator-driver chips and modules have significantly eased the design of peripheral IoT devices. Developers no longer need to spend time fine-tuning sensitive analog signal chains to optimize data conversion or to compensate for noise, temperature or drift in sensor subsystems. Smart sensors leverage integrated microcontrollers (MCUs) with built-in analog peripherals and radio transceivers to accurately convert sensor signals and wirelessly

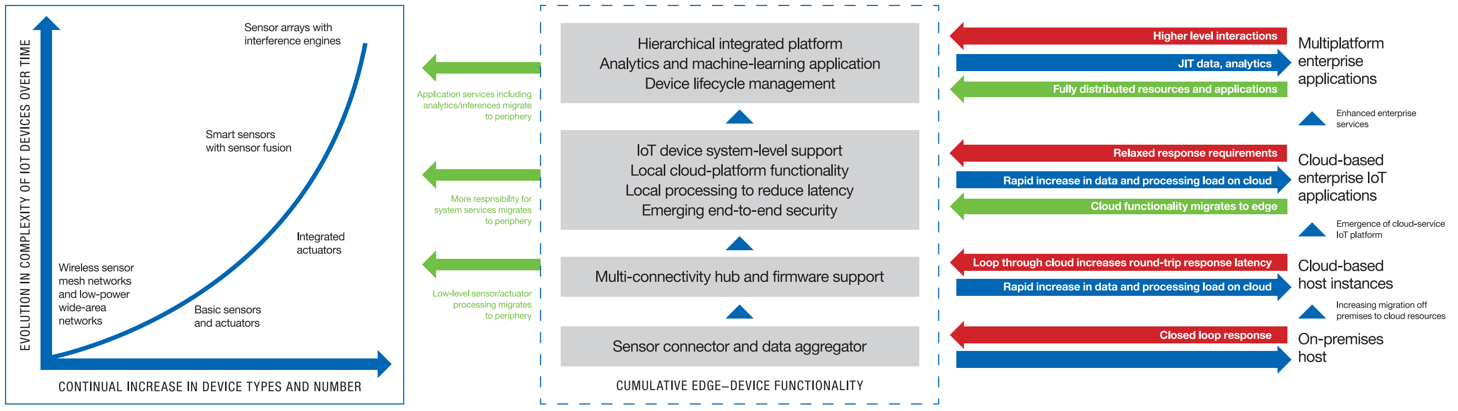

Sensor arrays with interference engines
Smart sensors with sensor fusion
Integrated actuators
Wireless sensor mesh networks and low-power wide-area networks
Basic sensors and actuators
EVOLUTION IN COMPLEXITY OF IOT DEVICES OVER TIME
CONTINUAL INCREASE IN DEVICE TYPES AND NUMBER
Application services including analytics/inferences migrate to periphery
More respnsibility for system services migrates to periphery
Low-level sensor/actuator processing migrates to periphery
Hierarchical integrated platform
Analytics and machine-learning application
Device lifecycle management
IoT device system-level support
Local cloud-platform functionality
Local processing to reduce latency
Emerging end-to-end security
Multi-connectivity hub and firmware support
Sensor connector and data aggregator
CUMULATIVE EDGE–DEVICE FUNCTIONALITY
Higher level interactions
JIT data, analytics
Fully distributed resources and applications
Multiplatform enterprise applications
Enhanced enterprise services
Relaxed response requirements
Rapid increase in data and processing load on cloud
Cloud functionality migrates to edge
Cloud-based enterprise IoT applications
Emergence of cloud-service IoT platform
Loop through cloud increases round-trip response latency
Rapid increase in data and processing load on cloud
Cloud-based host instances
Increasing migration off premises to cloud resources
Closed loop response
On-premises host

Figure 1: The simultaneous rise of both advanced IoT devices and cloud-based computing resources offers an extraordinary platform. (Source: Mouser)

transmit calibrated digital data. At the same time, the availability of larger memory arrays integrated into these devices have enabled manufacturers to extend communications support beyond integrated physical (PHY) and Media Access Control (MAC) layers found in earlier devices. Developers can take advantage of wireless MCUs and modules with complete communications stacks and even with the ability to support multiple connectivity options concurrently. Using these fully integrated communications solutions, developers can create IoT solutions more rapidly, focusing less on low-level operations and more on their application requirements.

The ready availability of diverse connectivity options in easy-to-use integrated devices serves as one of the fundamental enablers for large-scale IoT applications. The practical limitations of exploiting those options can sometimes make them seem more of a curse than a blessing for developers required to connect IoT devices to the cloud. Caught between the pressure of shrinking product windows and the complexity of diverse connectivity options, developers can find themselves forced to go for expediency over optimization. Rather than select the cloud-connectivity option that best meets performance requirements, developers can find themselves driven to settle for an option already in use at the target location or easily supported with off-the-shelf Wi-Fi routers, for example.

The evolution of edge devices has helped to relax many of the logistical constraints that limited selection of IoT device connectivity in the past. The same industry advances that enabled more connectivity options in integrated devices have simplified the ability to complete the connection to the cloud from IoT devices using the most suitable connectivity option. This greater freedom of choice has become particularly important in wireless deployments of power-limited IoT devices. Developers building extended-range IoT networks can more easily stay within an individual IoT device's limited power budget, using the optimal wireless technology needed to meet requirements for frequency, rate, and throughput.

"In this new view of the IoT, edge takes the motivation for local processing to its natural conclusion."

By taking advantage of the broad connectivity support of an edge device, developers can choose technologies such as mesh networks where inter-node relays extend the physical coverage of a network beyond the transmission and receiver range of individual low-power IoT devices. For connectivity across kilometers-long distance, developers can turn to sub-GHz low-power wide-area network technologies such as LoRa or SigFox. Using an edge device as a communications gateway for each connectivity option, developers can build optimized hybrid IoT networks that combine any number of non-IP-based subnetworks with IPv4- and IPv6-based subnetworks.

Besides its ability to support these connectivity options on the device side, edge devices provide an excellent platform for supporting wide-area networks with truly global coverage. Growing availability of low-power narrowband Long-Term Evolution (LTE) cellular options from chip makers, module suppliers, and cellular-service providers lets developers enhance edge devices with support for Cable and Telephone (CAT) M1 or Narrowband IoT (NB-IoT) connectivity. This additional option offers an increasingly vital feature for connecting remote IoT networks to the cloud or using cellular services to maintain availability in applications that must maintain data exchange even when primary connections fail to the Internet or private network.

Local Processing

Besides their use in enabling multiple connectivity options on the periphery, edge devices serve a critical need that has emerged with the migration from on-premises hosts to cloud services. From the viewpoint of a peripheral IoT device, the time needed to respond to external events can be considerably different for an application running on on-premises hosts compared to the functionally equivalent cloud-based application running on cloud services. Transactions between the periphery and cloud-based applications typically face longer communications times as well as additional delays related to security and other services required in a cloud service accessed through a public network.

In early IoT cloud-based applications, an incrementally greater lag between acquisition of data by sensors and the return of data processing results might have caused little concern. As users became more familiar with IoT applications, however, expectations have grown for even faster response to analysis of ever more complex data streams. In closed-loop industrial IoT applications, in particular, the longer path from IoT sensors to cloud-based data-processing software and then

[CONT'D ON NEXT PAGE]

finally back to actuators might extend response times well beyond control-loop timing requirements.

Like programmable logic controllers used before them in factory automation, edge devices move processing closer to data sources. By leveraging the processing capabilities of edge computing devices, IoT application developers can maintain the shortest possible latency between data generation and software response. As a result, developers can build industrial IoT applications that fully exploit the extensive capabilities of cloud resources without compromising the low-latency response times required in process-control loops.

The growing need for serving low-latency, deterministic-response requirements has motivated a further evolution of edge devices to better align with their role as active interfaces between IoT devices and the cloud. In this enhanced view, edge devices combine the capabilities of real-time systems needed to match the timing of real-world events with the features of general-purpose systems needed to support communications stacks and other applications-oriented requirements.

To serve these dissimilar requirements, advanced edge computing systems often use embedded hypervisor virtualization software to run separate partitions for each role. In these systems, code requiring low-latency, the deterministic response can run on a real-time operating system or kernel, while application code can run on a multitasking operating system such as embedded Linux. For more demanding requirements, developers implement the underlying hardware base using multiple MCUs or multicore MCUs, combining an MCU or core such as Arm® Cortex®-M optimized for deterministic real-time performance with an application processor or core such as the Arm Cortex-A designed for general-purpose computing.

Service Support

With enhanced local processing and storage, edge-system functionality further evolved beyond its earlier connectivity focus, now providing a more intelligent interface between IoT devices and cloud resources. To support more sophisticated MCU-based terminal IoT devices, this evolution, in turn, encompassed additional capabilities. Edge device functionality increasingly offers greater system-level support for IoT devices as well as more refined data-processing functionality needed to manage the growing flood of data created by these devices and demanded by higher level enterprise applications. Just as local processing provided the solution for low-latency requirements, it has allowed application developers to turn massive streams of raw sensor data into more manageable flows of useful information. Here, data service layers in edge devices provide application-specific services such as synchronizing disparate streams, performing sensor fusion, or identifying data of special interest for processing by local service handlers or by upstream services.

This elevation of edge capabilities from basic connectivity or even local processing has accelerated a new view of the edge as a more active participant in the myriad services underlying an IoT application. Indeed, enhanced connectivity of increasingly sophisticated MCUs in terminal and edge IoT devices has created a growing need for local services able to support a broad array of "non-functional" requirements including security, maintainability, and others required to ensure overall application health and quality of service.

Often, these requirements involve methods applied at each of the multiple vertical layers and horizontal partitions that define an application. For example, hardware or software flaws in IoT devices can present security vulnerabilities, weak connectivity protocols in IoT networks can provide an avenue for attacks, and complex cloud-based software systems that carry any number of security threats. Worse, IoT applications present a particularly attractive target. Using any of the multiple threat surfaces inherent in a connected application, an attacker can reach through the IoT network to strike at valuable enterprises resources typically tied in at the upper levels of an IoT application.

Like an IoT application itself, an effective solution to security and other non-functional requirements crosses multiple boundaries. For example, IoT security requires a multilayer approach. At the lowest level, IoT terminal and edge device security combine multiple low-level security mechanisms such as hardware-accelerated cryptography with higher level methods for secure firmware updates and secure boot. Building on this hardware root of trust, higher level protocols and policies provide aspects of security such as authentication, access control, and rights management to ensure that only authorized devices and cloud resources participate appropriately in the IoT application as a whole. The effectiveness of a solution to security and other non-functional requirements typically depends on its completeness: Each component is necessary for a robust solution, but none by itself is sufficient. To address this growing need for end-to-end mechanisms, methods, and policies, cloud providers have defined IoT platforms designed to better mediate the convergence of requirements of IoT device, middleware services, and application-level cloud services.

The Smarter Edge

With the emergence of IoT platforms, edge devices found a new role that transformed edge devices from essentially an add-on within the IoT hierarchy to a first-class member. Rather than provide limited processing for control-loops, device management, and data conditioning, smarter edge devices have become a central player in the sophisticated service-oriented IoT platforms offered by cloud-service providers. Here, edge devices have taken on some of the capabilities of each side of the interface between IoT devices and cloud resources, serving as a proxy for those capabilities to the other side. For terminal IoT devices, edge systems host a growing complement of local IoT platform services previously available only from the cloud itself. For the cloud, edge systems provided the cloud with access to virtual IoT devices containing configuration, state, and data that shadow the actual devices themselves.

In a sense, this new view of the IoT edge takes the motivation for local processing to its natural conclusion. IoT architects recognized that just as the inherent delay in cloud connections erodes response latency for time-critical control loops, sole reliance on cloud resources for device management was a losing proposition. Beyond its role in leveling data streams and reducing service complexity, a smarter edge device that acts as a proxy to both devices and cloud applications enables each to continue functioning despite failures on either side or even if the connection between the two becomes intermittent or lost entirely. If one or more terminal IoT device goes down for any reason, the edge system provides its shadow to the cloud application; if cloud resources become available, the edge system provides the equivalent cloud services most critical to IoT device operation. In each case, this proxy role goes well beyond that of a simple cache or lookup table. The edge device serves as an integral part of the IoT application as a whole.

This expanded view of the edge continues to evolve rapidly in response to the emergence of new methods designed to address growing demand for more powerful IoT capabilities. For example, the startling advances in machine-learning methods have allowed developers to respond to demand faster recognition of patterns of interest buried in all manner of data sources. Just as edge devices reduced response latency in control loops, they are speeding delivery of more complex information by moving inference engines from the cloud and closer to data sources.

The ability to execute machine-learning inference models and other sophisticated analytics algorithms at the edge allow IoT application developers to change the nature of what flows to the cloud. Rather than delivering data streams alone, smarter edge devices let developers deliver analytics and inference results without compromising their ability to deliver data just when and where needed. In this way, edge devices transform the mechanics of data and information flow in IoT applications, enabling developers to elevate their functionality and more efficiently knit diverse information sources into more effective enterprise solutions. M

Making the Case for the Smarter Edge: The Six Vs of IoT Data

by Stephen Evanczuk for Mouser Electronics

Developers face high data volumes, velocities, and variety found in "big data," along with the subtleties of data variances, vulnerabilities, and veracity. In response, there will be a greater dependence on a smarter edge to meld the IoT hierarchy into an effective end-to-end solutions platform.

The Internet of Things (IoT) is first and foremost about data but transforming that data into useful information is no easy task. Rather than idealized founts of perfectly formed data, IoT data pipelines can sometimes be chaotic, always changing sources that challenge each stage of their transformation to information. Along with demands tied to high volume, velocity, and variety familiar to any "big data" application, IoT data complicates the path to information generation with additional characteristics including variance, vulnerability, and veracity (**Figure 1**).

On their own, high-level approaches operating solely at the apex of the IoT hierarchy can find that dealing with these characteristics on their own is impractical—sometimes impossible. An increasingly important piece of the solution to these challenges lies in the evolution of a smarter edge that is able to decouple the details of data generation in the periphery, that is, from the abstractions that are essential to the high-level software and hardware, which ultimately generate useful information in large-scale IoT applications.

Volume

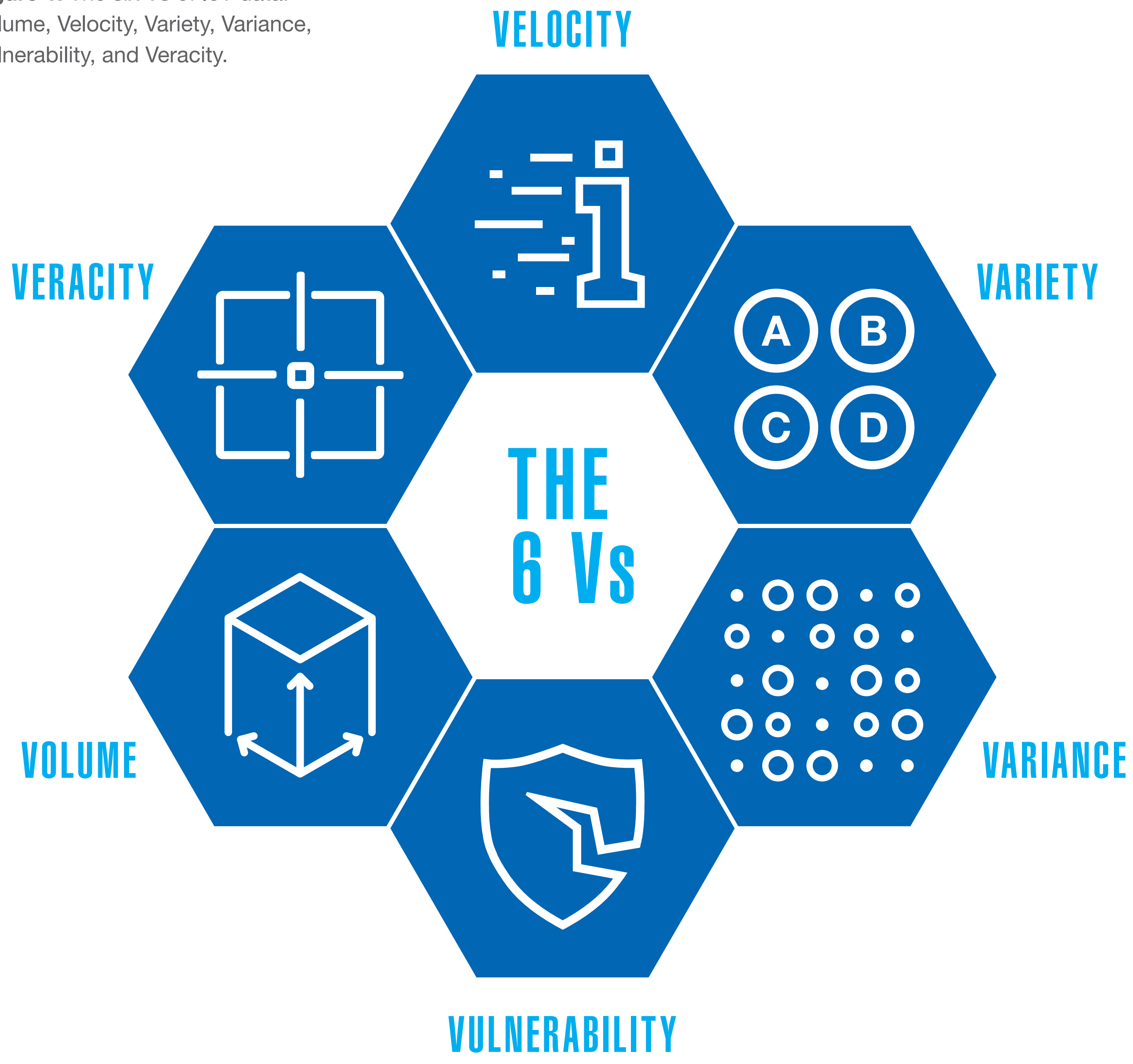
Enterprise IoT applications can involve IoT networks comprising hundreds of thousands of peripheral devices ceaselessly pushing data to data-hungry analytics and machine-learning algorithms. From a straightforward logistical point of view, the scale of these networks is enough of a concern. However, the number of IoT devices by itself tells only part of the story. The combination of advanced sensor technology and highly integrated MCUs enables developers to use a single IoT device to generate multiple streams of data. The availability of sensor fusion libraries allows developers to generate data beyond the raw data streams from sensors such as accelerometers, gyroscopes, magnetometers, and more. MCUs loaded with sensor-fusion firmware can form virtual sensors such as inertial measurement units that provide additional streams derived from that physical-sensor data.

While vast numbers of environmental and virtual sensors gradually raise the tide of data reaching IoT applications, growing interest in streaming video and audio data opens the floodgates. The rapid acceptance of highly accurate machine-learning algorithms means that deployments are no longer limited by the number of human eyes and ears available to observe these streams. Now developers can feed these algorithms with high-resolution data from low-cost image sensors and microelectromechanical system (MEMS) microphones to an extent that would have been impractical few years ago.

To sustain this volume of data, developers need to ensure that the underlying smart sensor systems operate at peak efficiency without further burdening the upper layers of an IoT application. Meeting this broad objective increasingly drives the need for smarter edge devices able to provide lifecycle support to a growing pool of sensors including commissioning new devices, securely updating their firmware, and retiring them when necessary.

Even without the additional administrative load associated with IoT sensors, the amount of data can be counterproductive for large-scale IoT applications. Not every application needs the full weight of data that an IoT network's sensors can provide. With a smarter edge device, developers can lower the volume of data traveling upstream

Figure 1: The six Vs of IoT data: Volume, Velocity, Variety, Variance, Vulnerability, and Veracity.



by running local inference engines or using data reduction methods to transform raw data to the resolution or update rate required by the cloud-based application.

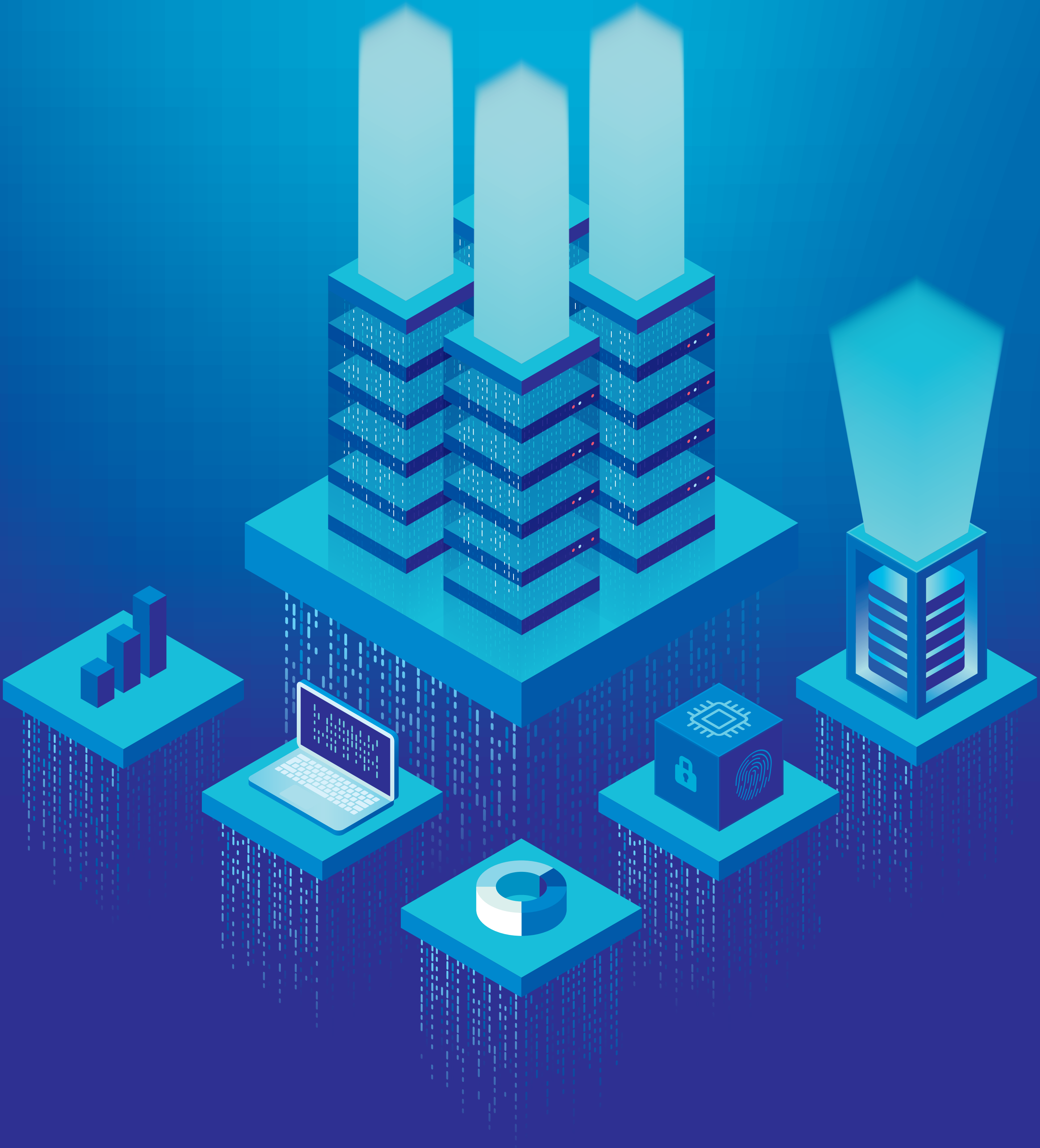
The local data transformation methods enabled by a smarter edge might not be optional to support essential high-level policies for security and privacy. For example, any application that touches personal identifiable information (PIL) will likely find a growing degree of difficulty as users exercise their right to privacy. Privacy experts use data minimization methods to filter out PIL that is not specifically required by the application or permitted by the PIL owner. As amply demonstrated in news headlines about security breaches, however, any data minimization performed in the public cloud leaves source data exposed as it traverses public networks and rests in a potentially vulnerable cloud storage.

A smarter edge device provides a perfect compromise: Application developers can process data streams that contain permitted PIL in the edge device and push appropriately minimized results to the cloud.

Velocity

Users' insatiable desires for better information levy a broad set of requirements on IoT developers. Besides teasing out more data

[CONT'D ON NEXT PAGE]



from more sensors, developers find themselves chasing demand for more accurate results along a more finely sliced time base. As its part, the IoT application is expected to respond more quickly to data arriving at higher and higher rates, not only from high-bandwidth streaming data sources but also from an expanding pool of smart sensors.

The impact of this unrelenting increase in data velocity sends ripples across the entire IoT infrastructure. In a more conventional cloud-based application, developers would need to continue scaling out the cloud infrastructure, adding more expensive high-throughput virtual server instances, input/output (I/O), and storage capacity. A smarter edge might not eliminate this demand for large-scale applications, but it could ease the need for those cloud resources to bear the unabated firehose of data arriving from the edge. Besides their ability to perform data reductions and minimizations locally, smarter edge devices can buffer peak-throughput demands, caching data locally on embedded solid-state drives. Also, developers can use these devices as smart load balancers, providing an application-oriented capability that is difficult to achieve purely with conventional-network load balancers typically placed on the public network side of cloud servers.

As with security and privacy, however, the changing nature of IoT data can directly impact fundamental requirements. In striving to respond quickly to the data input, the normal data flow through the IoT-application hierarchy may not be fast enough, and higher velocity data compounds the difficulty. Just as the smarter edge can be used to cull sensitive information from data streams, it can also allow developers to short-circuit the normal data flow. Rather than sending sensor data to the cloud for processing, developers can use the edge's local processing capability to meet requirements for low-latency response in closed-loop process-control environments. In this role, the smarter edge is fundamental to the success of Industrial IoT (IIoT) applications and enterprise-level manufacturing-execution systems.

Variety

The notion of a fast-growing pool of sensors naturally invokes images of a massive data firehose pumping more bits to the cloud, but IoT data streams are stunningly heterogeneous. The motivation behind the broad vision of the IoT would hardly find achievement in an application that continuously saw the same data regardless of its volume and velocity. If variety is the spice of life, it is the lifeblood of the IoT.

Although data variety gives rise to many of the same requirements mentioned earlier, it can impose a significant strain on the ability of an IoT application to respond to the varied needs of its different sensors and their operating modalities. Smart sensors have relieved some of the burdens developers faced in the past in working with sensors. For example, smart sensors typically provide self-calibration features and integrate compensation tables or algorithms—all capabilities that developers needed to perform implementations themselves in the past. Now, instead of setting bias levels, excitation currents, and other analog operating characteristics, developers work in the digital domain to programmatically set registers and configuration data in these sensors. For the microcontrollers (MCUs) paired with these sensors, a developer needs to provide more system-level capabilities and support for features like security updates to firmware and on-chip data, such as keys and certificates required for secure transactions.

In large-scale IoT device deployments, the myriad details and demands of peripheral device management present a significant challenge to IoT developers as well as the IoT infrastructure. By co-locating the necessary complement of device-oriented services for the application's devices, a smarter edge not only reduces the load on cloud resources but also simplifies the overall IoT application architecture. For this reason, cloud service providers have created IoT-specific service platforms designed to help deal with this complex problem. Indeed, a major functional component of these IoT platforms resides on the edge. Here, a smarter edge system supports the detailed transactions required to configure, update, and operate terminal IoT devices, allowing the cloud-based application to participate in device management at a higher level of abstraction.

Variance

The nature of IoT data becomes particularly interesting in its characteristics beyond the traditional big three of big data. In an IoT application, data variances emerge as an important refinement of data variety. While data variety is largely a static attribute of large-scale IoT deployments, variances in data from individual sensors is a dynamic quality that can reveal valuable information. Just as an IoT application fed with homogeneous data offers few, if any, real insights, an application

[CONT'D ON NEXT PAGE]

working with invariant data even from heterogeneous data sources largely provides a static snapshot of its target.

The ability to record such a snapshot at high fidelity and resolution might be enough in some cases. In applications as varied as consumer heart-rate monitors, process-control systems, or security management systems, steady-state data confirms the expected behavior. When the data departs from its steady-state values, the contingencies programmed into the application come into play, sometimes working to return to the steady-state values or sometimes simply watching for the next change.

The difficulty lies in recognizing if the variance reflects some significant activity or if it lies within a range of acceptable variation in the environment, the monitored process, or in the sensor signal chains. The ability to recognize the difference is critical for application success. The users of an IoT application will eventually lose confidence in any system that constantly issues false positives. At the same time, false negatives could reduce the timeliness and urgency of an eventual response. For example, the presence of data outliers could be the first sign of hackers testing the boundaries of the IoT application. Failure to act on these anomalous variations could result in an escalated attack or quiet penetration of connected resources.

Dealing with variances effectively requires both detection and interpretation—with functionality ideally placed as close to the data sources as possible. Using the local processing and storage capabilities in an advanced edge architecture, the edge-based software can track trends in data sensors and perform analysis with conventional analytics or machine-learning algorithms to determine if a variation in data crosses some threshold of significance. Although this kind of predictive "sensing" is only beginning to appear, its successful emergence in IoT applications critically depends on a smarter edge.

Vulnerability

Any connected application faces multiple security threats, and an IoT application lies perhaps more exposed than any other. Like any other connected application, an IoT application is vulnerable through the connectivity backbone that provides the foundation for data transfers and analysis. Unlike most connected applications, an IoT application comprises widely diverse systems ranging from the real-time environments running in terminal IoT devices to the general-purpose environments running in cloud and enterprise servers—or both environments running in edge devices. Add to this the volume and variety of peripheral IoT devices, and the result is an unparalleled collection of systems, each individually representing multiple points of penetration and in aggregate, presenting a porous gateway to hackers.

If this vast set of weak points provides the opportunity for hackers, their motive will be the prize available through an IoT application. As part of the enterprise infrastructure, IoT applications can tie into corporate data-lake systems, storage, and computing systems—providing a pathway to a black-hat community that has already demonstrated that it has the means to penetrate all manner of connected systems. Often lacking the resources required to harden security sufficiently, peripheral IoT devices can present themselves as easy targets for hackers looking to tunnel their way past network security and into valuable enterprise resources. Smarter edge devices are essential not only for enhancing the security of resource-limited IoT devices. They are, more broadly, vital within the IoT application hierarchy for extending high-level security policies across lower layers of the hierarchy, which often lack comprehensive protection.

In many ways, smarter edge systems provide a critical component in end-to-end IoT security solutions. At this level of security mechanisms, these systems use their local device management capabilities to update device firmware, provide lifecycle support to device security credentials, support secure commissioning, and more. For resource-limited IoT devices that are simply unable to apply security methods promptly, smart edge systems can identify sensor readings from these devices as untrusted but perhaps otherwise useable. Alternatively, smart edge systems can leverage their ability to support multiple connectivity options to reach these devices through short-range, non-Internet Protocol (IP) links, eliminating a direct pathway from the device to the network backbone and enterprise resources.

Beyond their support of low-level security methods, smarter edge systems can offer enterprise-level security policies for authorization, access, and privileges to the local level. As with device management services, these higher-level security

[CONT'D ON NEXT PAGE]

measures are an important feature of the IoT platform provided by major cloud-service providers. By hosting these cloud-related services at the local level, smarter edge systems can extend the policies of those cloud environments to the very periphery of IoT networks.

Veracity

The final characteristic included in this article is the most elusive of all, but it ultimately determines whether the users of an IoT application accept its results. In this sense, it is perhaps the keystone in the bridge between data and knowledge. In its most abstract interpretation, veracity describes the extent to which an IoT application provides a faithful representation of its target—whether that representation is a transformation of data through multiple analysis and inference engines or if it is simply a desired collection of data measurements. In more actionable terms, data veracity relates to the ability to validate the operation of each stage of the IoT flow from sensor signal input to knowledge generation. It addresses the following questions:

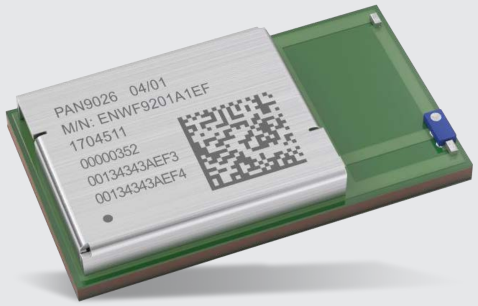
- Are wireless sensor devices operating correctly or introducing high-frequency radio signal artifacts in the data?
- Is the connectivity backbone transferring every packet in sequence or is the high volume or velocity of data eroding its quality of service due to congestion, delay, or even suboptimal protocol settings?
- Have hackers compromised devices or poisoned data streams with corrupt data?
- Are increased requirements overloading the edge and impacting the connectivity, local processing, or IoT platform services?
- Are machine-learning models correctly trained, or is overfitting causing their recall rate to plummet even with acceptable data variances?

Data veracity and its flip side, the application confidence level, are of course the objectives of every application development team. Because of their sheer complexity, IoT applications defy simple attempts to precisely quantify each of the myriad of requirements necessary to satisfy those abstract objectives. IoT application developers are more likely to fall short if they fail to account for the more evident challenges related to IoT data volume, velocity, variety, variance, and vulnerability. With the critical role that smarter edge systems play in meeting each of these challenges, the evolving capabilities of these systems provide developers with a vital resource for enhancing the veracity of IoT data flows and instill in a user a confidence in the data flow outcomes. M

Panasonic

PAN9026 Wi-Fi and *Bluetooth*® Radio Modules

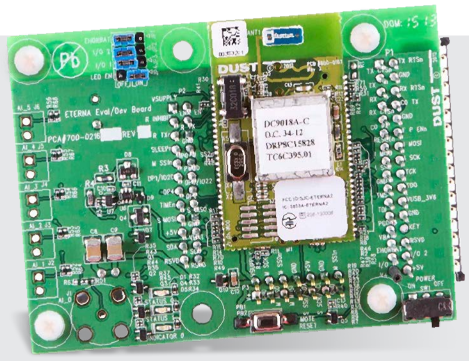
mouser.com/panasonic-pan9026-wifi-bt-modules



LINEAR TECHNOLOGY | NOW PART OF ANALOG DEVICES

SmartMesh® IP™ Wireless Solutions

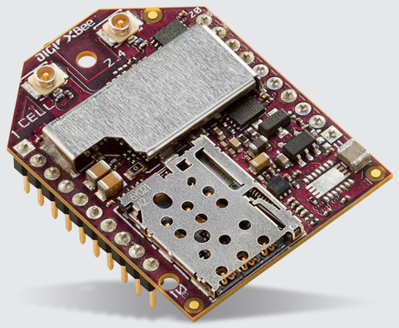
mouser.com/adi-smartmesh-ip-wireless-solutions



DIGI®

XBee Cellular
LTE-M Embedded
Modem

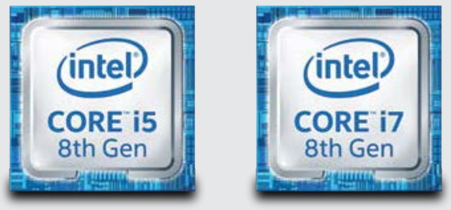
mouser.com/digi-xbee-cellular-lte-m



intel®

8th Generation
Core™ Processors

mouser.com/intel-8th-gen-core-processors



AI at the Edge Requires Balance of Capable, Flexible Hardware

by Stephen Evanczuk for Mouser Electronics

As the IoT expands, we are finding new ways to use emerging artificial intelligence (AI). But, AI hardware must offer balance and flexibility to address the multifaceted challenges found at the edge and open the doors for AI processing, to create the next great innovations in IoT applications.

Artificial intelligence (AI) methods have advanced with startling speed from a research topic to become a primary focus of the semiconductor industry. Already at work in smartphones and rapidly emerging in home devices, AI machine-learning techniques have rapidly gained attention for their ability to deliver accurate results using statistical methods. While a global race for AI chip dominance is only beginning, the use and availability of AI hardware accelerators at the edge of the IoT promises to leverage streams of data in providing application features that simply cannot be developed with conventional methods. For developers, the challenge lies in engaging these techniques with available hardware solutions and anticipating emerging AI-based architectures.

Rather than creating code to find events of interest, development teams can achieve remarkably accurate results using supervised learning methods to train models to identify specific patterns or using unsupervised learning methods to discover patterns buried in data streams. In particular, the use of inference models—that is, trained neural networks—has achieved spectacular results in classifying data. This combination of data-driven, "code-free" development and high accuracy of pattern recognition holds great promise for providing the "killer app" for the IoT, where vast streams of data might otherwise go largely untapped by overwhelmed software developers.

To fully realize machine-learning's potential in the IoT, however, machine-learning algorithms need to move out of the cloud to operate as close as possible to their data sources. The placement of machine-learning algorithms in the IoT hierarchy is critical. Too far out on the periphery, sensor data streams may be too few or too feeble to warrant use of a machine-learning model. In their role as mediators between peripheral IoT devices and the cloud, edge devices receive the mass inputs of multiple sensors. Furthermore, their emerging role in bringing cloud services closer to peripheral devices makes them an easy choice for migrating machine-learning models from the cloud.

Smaller Inference Models

Although cloud-based machine learning can take advantage of scalable computing resources, such as graphics processing units (GPUs), edge devices offer more modest resources. Even so, developers can already deploy machine learning on advanced edge systems by taking advantage of compression techniques that enable inference models to run effectively on general-purpose processors, particularly those with specialized extensions.

Since their dramatic demonstration of highly accurate image recognition earlier this decade, deep neural network (DNN) architectures, particularly convolutional neural network (CNN) architectures, have grown in memory size and computational complexity. While the CNN architecture that achieved breakthrough performance in 2012 required only eight layers, subsequent models quickly jumped in size, using dozens or even hundreds of layers to improve accuracy on the benchmark ImageNet data set. Large models can be hundreds of megabytes in size and require massive computational power not only for training but inference as well. Deeper models also mean longer inference latencies as dataflows through more layers of the neural network. By

[CONT'D ON PAGE 20]

sacrificing computational resources, model researchers could achieve truly impressive accuracy.

To perform an inference on more modest platforms, such as edge systems, AI researchers found they could reduce the precision of the internal calculations, resulting in smaller models with minimal impact on accuracy. Researchers found that they could (to a certain extent) dial in the required model size using this reduced precision approach along with other techniques. Implemented in compressed CNN architectures, such as MobileNet and SqueezeNet, these inference models can deliver respectable levels of accuracy even with minimal resources available on mobile devices and general-purpose platforms.

Still, these models are not the right solution for all applications. Users will not tolerate a "respectable" level of accuracy for many applications, and CNNs will not be the best algorithm for every application. CNNs are only one of many types of DNNs, and DNNs are only one approach among many algorithms that might be more suited to a specific problem domain. Techniques such as decision trees and ensemble methods like random forest algorithms have proven particularly effective in solving many of the problems found in the IoT, but other IoT problems might be best approached using Bayesian inference, support-vector machines, gradient-boosting trees, k-nearest neighbors, or other techniques among a vast set of emerging algorithms. The number of new algorithms and significant variations of existing algorithms is growing rapidly: Dozens of original research papers appear each day, offering new approaches to enhance accuracy, reduce complexity, and generalize their inference capabilities outside the original training data set. Few technologies have enjoyed the heady pace of advancement that machine learning has, and therein lies one of the significant challenges in building AI processing into edge hardware using AI devices.

AI Hardware Challenges

Advances in algorithms will continue to alter the balance between accuracy and resource requirements, and practical implementations will constantly seek to resolve the conflict between these two key characteristics in all AI-based solutions. With CNNs, for example, algorithm researchers significantly increased accuracy by sacrificing memory and processing resources—both already in relatively short supply in cost-effective edge designs. Furthermore, to meet real-time performance requirements typically found in IoT applications, the inference latency also becomes critical as does throughput for high-volume data streams such as video streaming. Also, for an inference engine to run within the relatively restricted environment of an IoT edge system, power consumption also emerges as a critical metric, so developers need models that minimize memory access cycles and processing loads. Finally, the development and deployment of an AI solution needs to work easily with existing (or at least compatible) workflows and enjoy an ecosystem capable of supporting life cycle requirements. Of course, all of this must contend with the unsettled nature of AI algorithms and the continued evolution of development environments able to support advances in machine-learning frameworks, high-level AI libraries, and lower level hardware-supported math libraries.

Useful AI hardware must successfully navigate this changing sea of requirements and capabilities, but available solutions necessitate significant compromises in some areas to reach useful levels in others. For example, general-purpose processors offer maximum flexibility while sacrificing the performance of inference models. Advanced general-purpose processors somewhat balance that trade-off, using single-instruction multiple data (SIMD) instruction sets to provide the degree of parallelism necessary in a machine-learning algorithm execution. Developers can find available SIMD processors that provide hardware support for neural-network functions in libraries designed to support inference models on these processors. For example, Intel's Xeon Scalable processor architecture helps accelerate operations in the Intel Math Kernel Library for Deep Neural Networks (Intel MKL-DNN). Similarly, SIMD features an Arm® Cortex®-A series and some members of the Cortex-M family's speed neural-network functions in the Arm CMSIS-NN—the neural network extension to the Arm Cortex Microcontroller Software Interface Standard (CMSIS) library.

The ability to accelerate machine learning directly relates to the hardware's ability to support the general matrix multiply (GEMM) operation, which is the predominant calculation in neural networks and other machine-learning algorithms. With their dedicated parallel execution hardware, GPUs excel at this type of operation but exhibit very large levels of power consumption.

Balancing Requirements

Beyond these devices, an emphasis on parallelism is reflected in each existing and emerging approach for building AI processing into edge hardware. Among existing solutions, advanced field-programmable gate arrays (FPGAs) provide one of the most effective solutions for implementing inference engines. Advanced FPGAs feature embedded memory and digital signal processing (DSP) slices, which together provide the essential ingredients of hardware-accelerated machine learning. Developers can use the FPGA fabric to implement resource schedulers and control logic. In turn, this control subsystem will optimize the execution of matrix calculations, using a combination of DSP slices for parallel calculations and embedded memory to efficiently access model parameters and store intermediate results.

For years, designers have exploited the flexibility and performance of FPGAs to accelerate compute-intensive algorithms not yet implemented in dedicated hardware. These same characteristics, along with their high-performance/watt capabilities, make FPGAs particularly well suited for implementing inference engines in edge systems. Indeed, machine-learning experts have focused on FPGAs as inference platforms in both research and emerging production environments.

Currently, significant effort focuses on deploying the FPGA-based inference in data centers, but the results of this effort are opening up a growing set of FPGA tools applicable to edge deployments. Today, developers can find support for their machine-learning developments in libraries, like the Intel Deep Learning Accelerator Library for FPGAs, the Lattice Semiconductor sensAI framework, and the Xilinx ML Suite, which are supported by FPGA device vendors. In each case, the libraries and related development tools provide the glue required to use the vendors' FPGA

[CONT'D ON NEXT PAGE]

devices as the hardware platform for running inference models developed on industry-standard frameworks such as TensorFlow, Caffe, and others. As part of their development workflow, these optimization tools in each environment let developers tune performance processes, perform quantization, prune trees, and employ other reduction methods such as merging separate model functions. Lattice Semiconductor provides specialized Internet Protocols (IPs) designed to accelerate an implementation of CNNs and binarized neural networks (BNNs)—a variation of CNNs that reduces model parameters to binary values with minimal impact on accuracy. In many ways, the pairing of FPGAs and advanced architectures like BNNs represents the core advantage of FPGAs, which offers developers the flexibility to pursue very new approaches with minimal risks. The downside is a more involved development process: Programming an FPGA is not as simple as writing code for a SIMD processor.

The variations of DNN architectures, such as BNNs, reflect a trend that will continue to play out as the industry works to perfect machine-learning deployments. Besides reducing memory requirements, BNNs simplify the math. Rather than needing to execute a computationally expensive GEMM operation, the hardware platform only needs to perform bitwise XNOR operations, which are all operations that execute rapidly on FPGA platforms, typically along with bit shifts.

Hardware Optimization

As AI researchers dive deeper into the details of algorithm implementations on hardware, they will find similar opportunities to tweak algorithms to improve a match to hardware. Conversely, this deeper understanding can also offer insights into hardware optimizations such as Intel's Flexpoint, which plays a central role in its Nervana™ Neural Network Processor (NNP). In studying what AI researchers call the "unreasonable effectiveness" of reduction techniques for inference models, Nervana engineers noticed that parameters of a specific matrix, or more correctly a tensor, stayed within a given dynamic range during training. Because a Nervana NNP's training fails when applying the kind of quantization used in reduced CNNs (and much fewer BNNs), employing Flexpoint is the most effective way for the Nervana processor to perform tensor operations as fixed-point operations rather than computationally expensive floating-point operations. Using an algorithm called Autoflex to manage these shared exponents in a tensor, training can proceed with fixed-point-like performance while retaining floating-point precision, opening the door to the highly valued prize of performing model training as well as inference in the field.

Optimizations on the algorithm side like BNN's bitwise calculations and on the hardware side like Nervana's Flexpoint will likely continue regardless of what happens in the imminent appearance of specialized AI chips. For these devices, engineers are combining techniques learned in massively distributed systems design with those learned in microsystems design and semiconductor design. Along with algorithm advances, this confluence of multidisciplinary design methods will begin to deliver chips that more successfully support the requirements of accuracy, resource utilization, power, flexibility, and integration—both individually and *in toto*.

This next stage of hardware support for AI is just beginning, but certain architectural principles are already emerging. Among these principles are evolving techniques for memory optimization and massive parallelism, which is not surprising. Using approaches remotely similar to processor cache hierarchies, advanced AI architectures are reducing both the number and duration of memory cycles by employing integration processing resources with wider, deeper memory hierarchies. For example, IBM researchers have described an AI chip that, among other features, includes "scratch pad" memory to reduce the need to access memory for model parameters and intermediate results. For its intelligence processing unit (IPU), Graphcore uses a new twist to a decades-old, system-level distributed memory approach, called bulk synchronous parallel (BSP) computing, to optimize model processing.

Of course, the success of machine learning finds its origins in GPUs, and the success of GPUs rests on their massive hardware parallelism. AI chips will build on these lessons learned in GPU architectures that exploit parallelism and in algorithm designs that optimize the replicated execution engines. For example, the latest version of Nvidia's Tensor core supports lower resolution integer values, keeping pace with advances in model optimization methods.

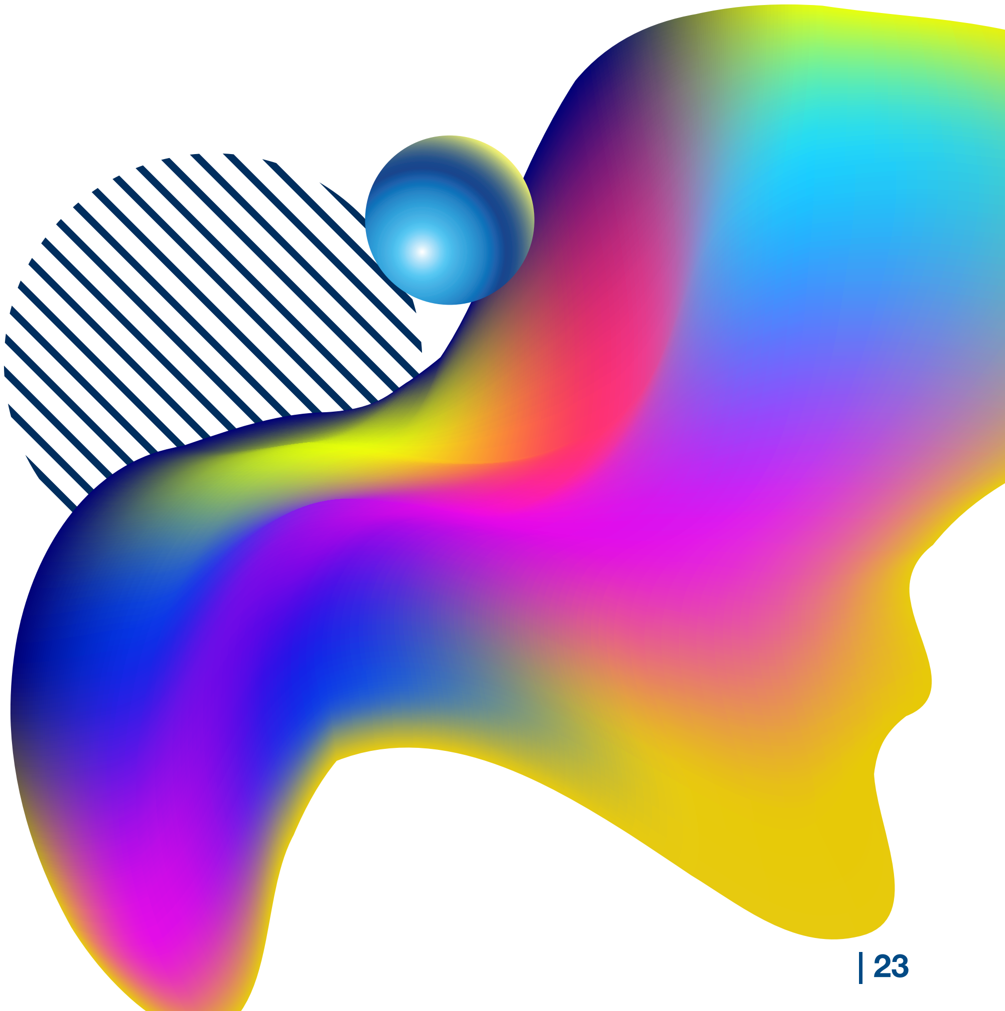
The performance of AI chips rests on the ability to replicate processing cores like Nvidia's Tensor in very large numbers—that is, without succumbing to performance degradation issues related to route ability, path timing, and interconnection planning in physical chip design. The combination of silicon design tools and AI IPs from Cadence, Synopsys, Imagination Technologies, and other IP vendors offers an effective base for highly differentiated AI devices. In fact, with its Deep Learning Accelerator (NVDLA) IP, Nvidia has entered the AI core arena with an open source, modular architecture designed to accelerate an inference and that even features a small-model design specifically intended for the IoT.

Conclusion: Flexible Solutions

The ability to support parallel matrix operations lies at the heart of emerging AI chips, but a third principle addresses both the reality of fast-changing algorithms and the need for product differentiation. Companies depend on their development prowess to differentiate products, largely through a code base with differentiated features that are built according to the company's core competencies. In providing a "code-free" approach, "standard" machine-learning architectures, such as CNNs, ostensibly deliver non-differentiated solutions. In reality, machine-learning developers aggressively modify standard model architectures, adding new layers to the pre-trained models used for transfer learning or for embedding complete models as pre- or post-processors, particularly in feature extractions or classifications in novel architectures. A specialized AI device tied to a particular architecture would complicate this practice, much less the overarching goals of product differentiation.

Recognizing both this need and the changing nature of algorithms, AI chip designers are offering two types of engines: Replicated "fixed-function" engines that are optimized for matrix operations, which are necessary for any machine-learning algorithm, and replicated programmable engines, which are designed to deliver the flexibility essential to accommodate algorithm advances and differentiated solutions. Arm's ML processor architecture illustrates this trend. A component of Arm's AI-focused Project Trillium, the ML processor architecture integrates multiple fixed-function, matrix-execution engines with multiple programmable engines. Dedicated static random access memory (SRAM) addresses the need for shorter memory-access paths to help accelerate an inference in each engine instance.

AI chips built with the Arm ML processor core and other architectures will begin to address the combination of challenges associated with an inference at the edge. Even these devices are only the beginning, focusing largely on proven CNN algorithms for proven markets such as those of automotive driver assistance systems (ADASs), surveillance, and the emerging class of applications built on low-cost, high-resolution image sensors. Even so, these AI chip combinations of programmability and AI acceleration makes them a compelling platform for building AI processing into the more sophisticated edge systems in IoT applications.



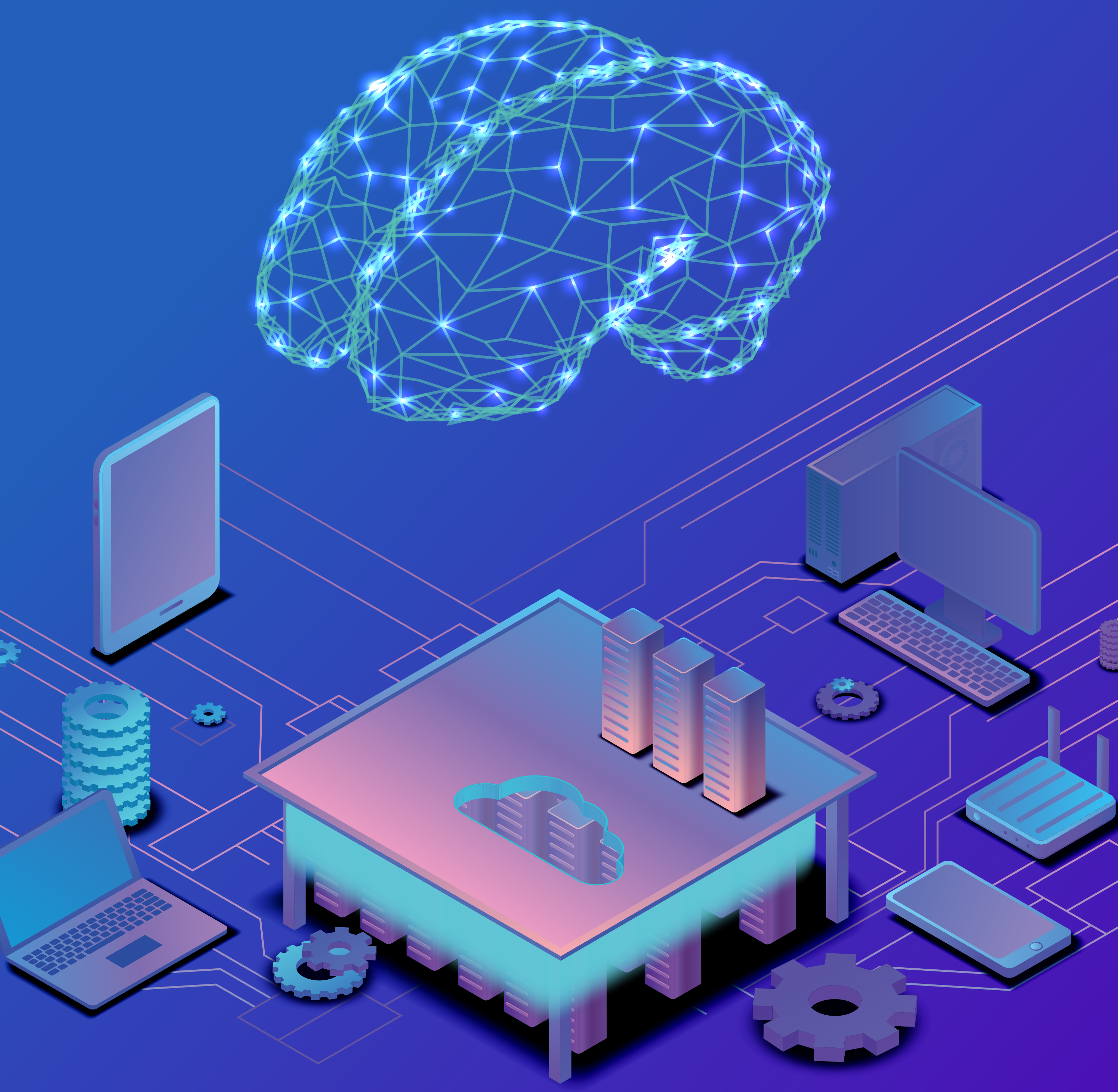


Figure 1: Artificial intelligence (AI) increases find utilization in a wide variety of applications. (Source: Mouser)

AI Processing for IoT: Where Clouds Give Way to a Smart Edge

by Paul Golata, Mouser Electronics

This article examines how AI processing capabilities are moving from the cloud into the smart edge and are providing game-changing, intelligent ways to enable tomorrow's IoT.

Technological breakthroughs in the area of semiconductor processors have enabled a host of artificial intelligence (AI) capabilities within the cloud computing domain. AI brings intelligence to devices to allow them to behave in more reliable or performative ways. Because it is in the electronic domain, theoretically AI is not bound by human biological limitations regarding capacity or response times. The cloud provides location and necessary processing power to handle large amounts of data and output innovative solutions that were previously unobtainable. Processors, accelerators, graphics processing units (GPUs), and field-programmable gate arrays (FPGAs) are semiconductor products that provide the necessary computational-processing power to the cloud (**Figure 1**).

For tomorrow's applications, it is no longer sufficient to keep AI only in the cloud. The Internet of Things (IoT) requires that more and more of the processing work happens closer to the end nodes—the location of data (sensors) and actuation (control). The ability to collect data, store it, and then perform the analysis is providing a myriad of new ways to do business. The demand for low-latency, real-time decision-making and response, which is imperceptible to humans (<0.05s), demands that AI processing moves from the cloud to a "smart" edge to transform our present business processes and products to those that the market wants tomorrow.

The edge exists in several contexts of which the IoT layer is one application. The edge is intermediate between the end-node device deployment and the top-level cloud-computing system. As an intermediate layer, it is now being provisioned with real-time AI capabilities, enabling applications that do not require computational power or cannot sustain the latency of cloud computing. The edge is becoming smart because more and more it is incorporating intelligence in the form of edge computing AI. Edge computing AI may experience use in various locations and stages between the cloud computing layer and the end nodes layer including the location of gateways, access points, and metro edges.

Inferencing at the Edge

Motion pictures provide the illusion of reality because multiple frames pass before the eye faster than the eye responds, creating the illusion of ongoing motion and no discontinuity. IoT applications, particularly those directly interacting with humans, require the same kind of ongoing continuity to avoid wait periods. Humans want the devices that they

[CONT'D ON NEXT PAGE]

Figure 2: AI and ML depend on high-performance semiconductor processors. (Source: Mouser)

electronically interface with to work with them in real-time.

To enable future IoT applications, including autonomous vehicles, real-time decision-making is required. Data at the edge moves to the cloud, and what comes back is features or responses to that data. In that small moment of time, the decision of whether a particular object in front of a traveling vehicle is a blowing piece of litter, a ball, another vehicle, or a person may arrive back at the vehicle too late. Voice-enabled applications now found in many homes respond within the response time expected of a normal human conversation. Detecting errors and defects within the industrial contexts may help prevent accidents and hazard occurrences. These specific applications are only the tip of the iceberg among numerous IoT applications coming along in the future that will call for immediate, real-time decision-making.

One method to make this happen is to enable inferencing at the smart edge layer instead of at the higher-level cloud computing layer. Inferencing means to draw or reach a conclusion based on the evidence by way of reasoning. AI at the cloud-computing level can process large amounts of data over time and draw conclusions from the patterns it discerns. It is possible to conceive of ways of pushing these categorized conclusions down into the smart edge layer, so that upon initialization, the smart edge has a wide variety of highly intelligent categories from which to start its computational processes. The smart edge effectively provides a shortcut, greatly expanding the capability for the smart edge to decide something faster than a human's biological response time. To enable this to happen, the smart edge must be able to process fresh data against these categories and quickly perform inferences towards a correct conclusion. Smart edge inferencing happens through machine learning, an application approach utilizing AI. Machine learning (ML) is a process or set of rules that occur through calculations or other problem-solving operations to empower the extraction of structured categories and representations from received input data. Processors, accelerators, GPUs, and FPGAs at the smart edge can be programmed to employ algorithms that draw these inferences. The cloud layer supports ML by ensuring that the smart edge always contains the most appropriate algorithms, based upon the computational power it exerts upon the vast raw data it studies and processes to gain experience. The cloud layer is the best layer level to train ML algorithms. The ML algorithm training from the cloud layer is the informational knowledge that passes on to the smart edge to employ. Trained ML algorithms at the smart edge enable the smart edge to process the real-time data it receives in a structured manner and to compare this data to data models: This process is a recent development called federated learning that is an alternative to centralized training.

Training ML algorithms at the cloud layer requires the highest performance processors, such as GPUs. However, these higher performance processors may not necessarily be a requirement at the smart edge. The reason is that the highly intensive computational processes occur at the cloud layer before moving to the smart edge. The question that engineers must answer is what level of computational quality metrics are necessary for learning. Also, determining how quickly the model needs updates with new data provides input regarding where processors should reside.

This deductive methodology allows for the proper optimization of a necessary performance level relative to power efficiency for action at the edge: Thereby enabling a greater variety of intelligent applications to perform without the necessity of the most costly, power-consuming processing chips (**Figure 2**).

One common example of an AI and ML performance at the smart edge of IoT is in image and video analysis. In this example, AI by way of ML processes a large amount of raw data and extracts usable data content to assist with future decision-making. When data arrives by way of new observations, an ML model processes this data to produce a classification or decision (**Figure 3**).

Processing Chip Architecture

The smart edge generates, handles, and utilizes large amounts of data. Due to this large amount of data, high-computing processors with excellent efficiency are desirable. Arm technologies enable the world's most popular AI platform—the smartphone—along with ML features like predictive text, speech recognition, and computational photography. Within Arm's line of high-performance, 64-bit (with a full 32-bit compatibility), Armv8-A processors are several devices such as the multicore Arm® Cortex®-A53 and Arm® Cortex®-A72. World leading semiconductor firms may employ these Arm cores coupled with vector-processing engines—in the form of a system on chip (SoC)—to support AI processing at the edge. An important aspect of SoCs is that they frequently integrate with things commonly found outside of the processor (such as accelerators for applications, busses, and interfaces, etc.). By this extended integration, SoCs provide flexibility and scalability, allowing designers to match their connected end node devices with the data, security, and performance characteristics that are essential to their IoT systems and applications at the smart edge.

Processing chip architectures in the smart edge want to take advantage

Figure 3: AI and ML in the smart edge can analyze videos and images to make decisions. (Source: Mouser)

[CONT'D ON NEXT PAGE]

of high, next-generation speeds (≥100Gbps) with excellent packet-processing abilities. Standard- and open-programming models that employ software-aware architecture frameworks make it easier for designers to configure the product to match their specific network requirements. By providing a core-agnostic architecture, designers can select the optimum core for their particular application. This concept takes advantage of the multicore trend and extends it by allowing an increased performance through the incorporation of either related cores or diversified cores. This smart edge computing generally contains provisions that support cybersecurity requirements that are inherent in IoT applications throughout their entire lifecycle. They also support virtualization—the abstraction of computing resources—permitting lower costs and complexities to prevail in designs: Virtualization achieves these benefits by treating computational and storage resources as separate entities, then redirecting and managing them for optimal utilization (**Figure 4**).

Another processing chip architecture is the x86 complex instruction set computer (CISC) microprocessor. The x86 architecture has been around for about two decades. Similar in architecture to central processing units (CPUs) found in desktop computers, they incorporate a more advanced feature set to meet the requirements of workstations and networks, making them suitable for smart edge applications. They work in smart edge applications to provide computational processing power, enhanced connectivity, and storage on a high-speed product that does not compromise on keeping every piece of data secure. The x86 architecture allows for an intelligent workload placement, a low latency, scalability, and extreme responsiveness. These processors contribute to optimizing performance, providing analytics, and offering accelerated data compressions. These microprocessors come in several performance levels, with the highest level of performance being suitable for demanding smart edge applications including real-time analytics, AI, and ML. They perform well when provisioned to give real-time AI-inferencing results at the smart edge. They come with up to 28 CPU cores and can support up to 12TB of address space.

Conclusion

The smart edge is emerging from the clouds. AI processing is enabling the smart edge to do a variety of tasks, which were previously relegated to the cloud computing layer. The demand for flexible, low-latency, real-time IoT solutions is part of the ongoing requirements that are now getting addressed at the smart edge. The ability to constantly adapt and provide data collections and analysis at the edge will allow AI and ML to provide better drive, business transformations, and performance. By managing information from a series of end node devices and applying ML algorithms on data business assets, IoT applications will be connected in a real-time, optimized way to decrease costs and increase value deliverables to the customer. The future of IoT includes soaring into the clouds only when necessary, for much of what we will accomplish will get performed at the smart edge. M

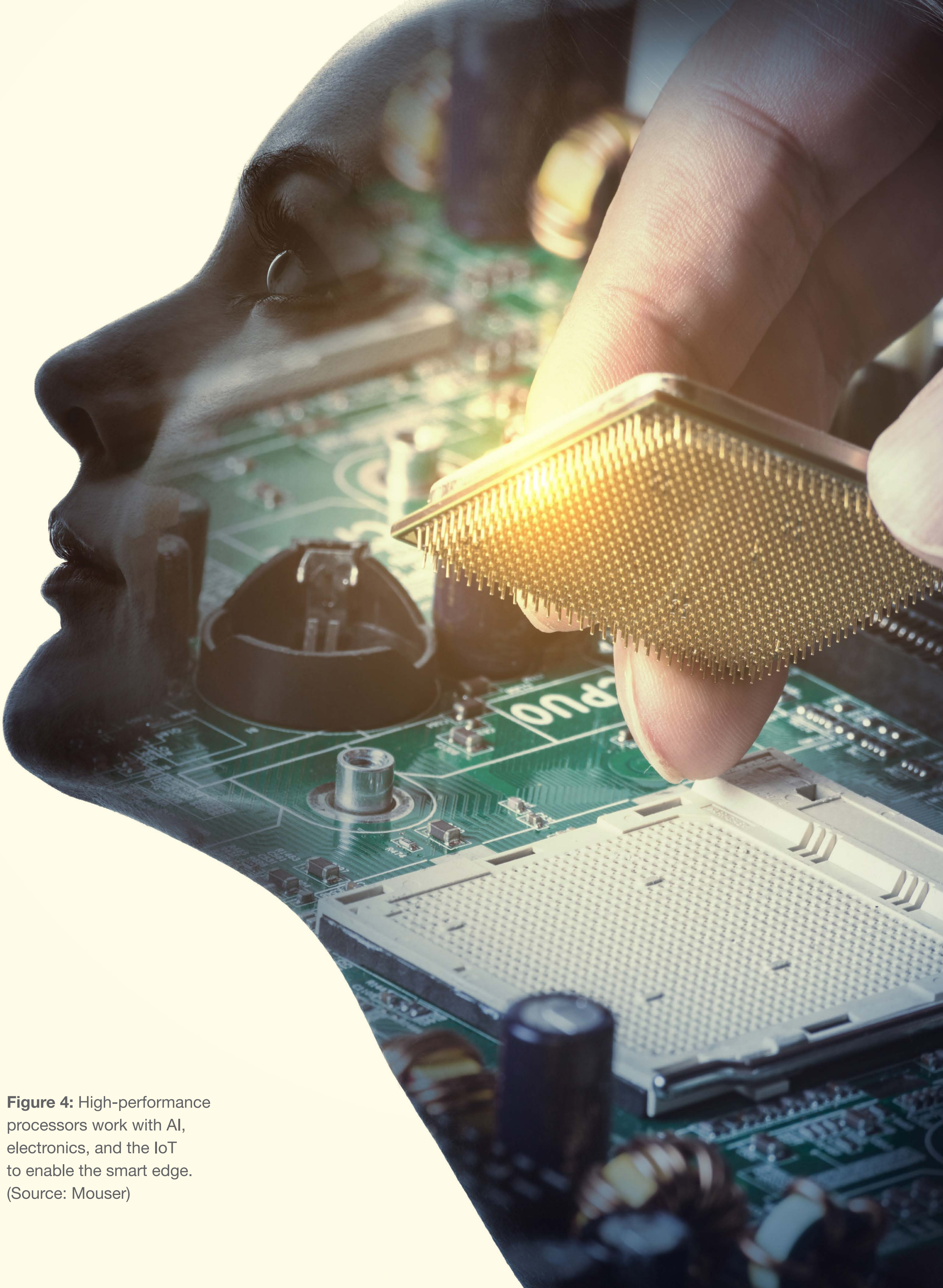


Figure 4: High-performance processors work with AI, electronics, and the IoT to enable the smart edge. (Source: Mouser)

Could the Edge End the Connectivity Wars?

by Bin Jiang, Chaofan Ma, Huifang Xu, and Houbing Song for Mouser Electronics

Connectivity wars stem from four common causes of data obstruction: Data differentiation from multiple devices, the data blocking phenomenon, the absence of isolation for data security, and reliability inconsistencies. This article examines how the edge can address these issues and end the connectivity wars.

As connected equipment, devices, and appliances proliferate, they are flooding networks with growing volumes of incompatible data. A house is a good example. Connected consumer devices like media centers and sound systems, climate control systems, household appliances, health monitors, interactive devices like Alexa, in addition to tablets, personal computers (PCs), smart phones, and cars are all pushing networks to their limits. These competing data streams can introduce latency into time-sensitive data, which degrades the user experience. The same thing can happen in an industrial setting where many more kinds of devices are competing for network bandwidth. In those situations, latency caused by data streams crowding each other out can disrupt a critical process. As connected devices become smarter, they generate more data, and this aggravates the problem.

Network service providers are beginning to use edge computing techniques specifically designed to optimize network performance under these conditions. Edge computing is a network architecture that places data processing capabilities as close to data sources as possible. When applied to network traffic, edge computing can optimize traffic to eliminate latencies and assure an optimum dataflow.

This article discusses some of the technical challenges presented when different types of data share the same network bandwidth, and how edge computing can improve network performance.

Examples of Edge Computing in Different Wireless-Connectivity Scenarios

In many application scenarios, edge computing is an appropriate tool to achieve good network connectivity. Here are three examples:

Local Video Streaming Distributaries

Under certain demand situations, large-scale local video streaming is inevitable. Such services contain wireless, high-definition (HD) video cameras; local multistream video pushing; and real-time multiangle viewing. In this scenario, wireless connectivity is a challenge.

Edge computing can add high-definition and low-latency video streaming, helping users choose and helping businesses deliver a new multichannel experience. More importantly, edge computing can improve network efficiency and reduce network backhaul pressure. While providing multichannel, full-angle, and HD video, edge computing reduces the live broadcast delay. Edge computing can also help deliver a wonderful video experience to audiences, even in the face of fast-changing scenes.

Virtual Private Networks for Enterprises

In an enterprise virtual private network, the base stations are directly diverted to the intranets within the enterprise. By integrating the intra-enterprise communication platform, edge computing can meet the requirements of network security, low latency, and big traffic. Based on these characteristics, this kind of network attracts enterprise customers. Also, it can combine the Internet of Things (IoT) to do target tracking and video surveillance.

All confidential data within the enterprise can bypass the public network, which enables the enterprise to carry out video conferencing, online learning, and other large-scale, real-time applications. At the same time,

[CONT'D ON PAGE 32]

the edge can make use of the mature, huge ecological chain of public networks; reduce the cost of related terminals; and integrate many enterprise applications through the open architecture application-programming interface (API). These capabilities increase technical flexibilities available to the business.

Real-Time Throughput Identification for Each Terminal

For complex connectivity, throughput identification is another challenge. To enhance the user experience, mobile edge-computing can provide a real-time throughput identification for each user's connection: During this process, uplink data transmits through the user interface without additional signaling interaction, effectively improving the real-time connectivity of Internet content sources. In this way, it can enable the utilization and optimization of end-to-end network resources more effectively.

Edge computing can create new value chains and ecosystems by improving connectivity. In this way, it can enable the various roles of an entire industrial value chain to benefit from a better cooperation. Based on the same network-connection mode, the entire industry chain integrates telecom operators, equipment manufacturers, information technology (IT) vendors, and applications and content providers who manufacture chips and servers.

Obstacles to Good Connectivity and Dataflow

To understand how edge computing can end the connectivity wars, it's useful to understand four common causes of data obstruction in wireless connectivity:

Data Differentiation from Different Devices

Unstructured data formats vary among different devices, which challenges the unification of data interfaces. In general, distinct network systems have uniquely diverse communication protocols, thus hindering the connection, which is also an important reason for the wireless connection war. Insignificant differences in communication protocols between dissimilar devices can have a significant impact on the entire network, thereby making it impossible to connect smoothly. Network edge-processing can greatly improve this situation.

Data Blocking Phenomenon

With the continuous increase of data in the IoT, successful dataflow is another challenge. Data blocking will inevitably cause problems for wireless connectivity. At present, there are many kinds of smart devices with sensors connected to networks. The International Data Corporation's (IDC's) statistics show that more than 50 billion terminals and devices will be networked worldwide by 2020. This kind of connected equipment generates enormous amounts of data that requires processing. According to IDC's prediction, 40 percent of the data in 2020 will need to be analyzed, processed, and stored on the edge of the network.

Isolation for Data Security

Network security is a critical issue, and a key piece of this security issue is related to connectivity. This critical issue includes the incorporation of user password authentications, user access controls, and mode controls. Good connectivity helps to prevent computer viruses and encrypts core data to create a complete network-connection environment. However, in many cases, the hidden danger of data security makes connectivity impossible, introducing connectivity conflicts between different devices and affecting how the entire IoT works.

Reliability of the Wireless Connectivity

The existing wireless connection technology used by consumer devices does not meet the performance requirements of industrial and healthcare systems. Different arrangements for security, accuracy, and time sensitivity of these systems increase the demand for reliability. The honeycomb system can almost meet these performance requirements, but it is often inappropriate regarding battery, cost, and data-transmission requirements. Edge computing can reduce the traffic of data transmissions in wireless connections and, consequently, save resources to improve the reliability of the wireless system.

Edge-computing techniques can go a long way towards resolving these dataflow obstacles in wireless networks. In the next section, we take a closer look at how edge computing can make a difference.

Using Edge Computing to End the Connectivity Wars

We have looked at application scenarios where edge computing can play a role in improving dataflow in wireless networks, but how specifically is this accomplished?

Let's take a closer look at the technical strategies employed by edge-computing techniques:

Edge Computing Can Avoid Wireless Network Pipelining

Edge-computing platforms provide open API interfaces for partners to develop applications. These open interfaces allow partners to drive new business models by applying innovative incubation patterns, actively fostering application partnering ecosystems, and supporting the full application development life cycle. Innovation incubators provide partners with the framework they need to work efficiently so that newly developed applications can be incorporated into the application system quicker. An edge-computing platform avoids wireless-network pipelining, which reduces staged workloads. These reductions then free up network resources.

Edge Computing Can Improve Network Slicing Technology

In network slicing technology, uniform connectivity is very important. It splits the existing physical-network into several independent, logical networks, providing specific operations for differentiated services. For example, a software-defined network (SDN) can define an open interface between an operation and the data it interacts with, thereby revealing the functional changes within a set of network slices. However, practical commercial applications still face many challenges.

Most current networks use the same access architecture, but application requirements usually necessitate an independent access mechanism and protocol stack for each application.

One of the main technical characteristics of edge computing is low latency. To achieve low latency, mobile edge-computing must support a very demanding flow of data traffic. That makes mobile edge-computing the key technology for ultra-low latency slicing. By applying edge computing, the purpose of network slicing technology will be extended from simply splitting out multiple, virtual end-to-end networks to splitting out virtual end-to-end networks with different high-delay requirements. The result of this is items requiring real-time processing that are coordinated separately and distinctly from those that do not require real-time processing. This distinction will assist in the development of an improved network slicing technology, leading directly to improved connection performances.

Edge Controller/User Separation Improves Wireless Network-Connection Compatibility

Current networks are facing exponential growth in traffic demand, so the use of a broader spectrum has become a new method to expand network capacity. However, compared with the lower-frequency band, the high-frequency band is liable to suffer serious propagation losses.

By separating the control layer from the user layer, a user gateway can synchronize independently in moving to the edge, thus naturally solving the related data-security problems during connectivity. Therefore, as one of the trends of fifth generation (5G) technologies, controller/user (C/U) separation is also the key edge-computing technology, which can provide a solution for network security. Specifically, edge devices take into account data collectors and owners respectively. For example, data collected by mobile phones will be stored and analyzed at service providers while preserving edge data and allowing users to own it. This will be a better way to protect data privacy. In this mode, the establishment of two discreet layers will allow for the full protection of user data within a specifically established, secure, and reliable network connection.

Edge Computing Meets Low Delay Requirements (As Is Expected with 5G)

Under the requirements of "big data" and high connection densities, ensuring the presence of low latencies and low-power consumptions is also very important. For example, there is a business goal of achieving 1ms of end-to-end delay in a 5G network so that it can support the needs of industrial controls and other services. However, designers cannot optimize the current mobile technology enough to meet this goal.

Long-Term Evolution (LTE) technology can improve the empty throughput by ten times but can only optimize the end-to-end delay by three times. The reason is that the network architecture is not fully optimized and thus is now the bottleneck of service delays though the empty port efficiency is experiencing a great level of improvement.

With the progress of mobile edge-computing, technical characteristics and industrial cooperation requirements are now functioning in many practical

[CONT'D ON NEXT PAGE]

operations, greatly improving a user's business experience. Edge computing now resides on the mobile edge, and wireless networks are integrating effectively. Because application services and content deployments are now on the mobile edge, their connection will be much easier. The result is that the forwarding and processing time of a data transmission will decrease. Additionally, end-to-end delays will decline, meeting the low delay requirements. All of this will be achieved while reducing required-power-consumption needs.

Ultimately, local computing services on the edge of wireless networks will undoubtedly meet the challenges of integrating different services among vendors, IT solution providers, and developers within heterogeneous environments. Edge computing will make use of access to these wireless networks to provide services necessary for telecom users and cloud computing. In this way, it can create a high-performance, low-latency, and high-bandwidth service environment, and consumers can enjoy a continuous, high-quality wireless-network experience.

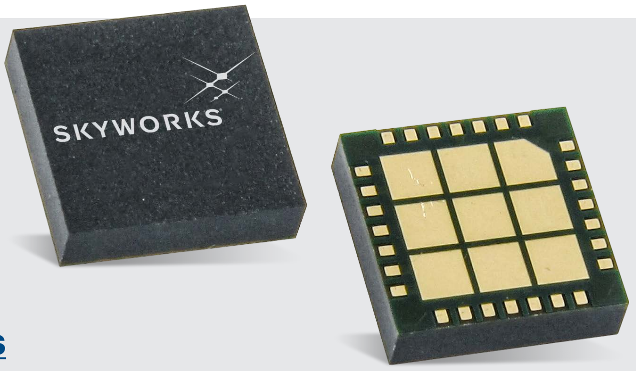
Conclusion

Difficulties arise in handling digital data when any technical obstacle prevents a barrier. This article

SKYWORKS

IoT Front-End Modules (FEM)

mouser.com/skyworks-iot-front-end-modules



TEXAS INSTRUMENTS

CC3120 SimpleLink™ Wi-Fi Network Processors and Modules

mouser.com/ti-cc3120-simplelink-processor

DELL

Edge Gateway 3000 Series

mouser.com/dell-edge-gateway-3000

discussed how the obstacles of different data types, clogged data pathways, the need to isolate data for the purposes of cybersecurity, and inconsistently reliable wireless connectivity all war against success.

Unified data connections and aggregations are the basis for the development of wireless communications and the IoT. Facing diversified standards and heterogeneous technologies existing across industries, the data from different connectivity sources often cause data blocking. Additionally, different wireless-connectivity options, which many diverse devices use within the IoT and which aim to coexist in the same networks, inevitably create data conflicts. To avoid the pipelining of connectivity, it is very important to enhance connectivity through a deeper integration, especially in mobile networks. Integration and interoperability across vendors and domains are essential.

Edge computing effectively integrates wireless networks and Internet technologies, while also introducing computing, storage, processing, and other functions into a wireless network. It builds an open platform to implant applications and opens an information exchange between the wireless network and service servers through wireless APIs. For business level connectivity, edge computing can provide customized and differentiated services to industries, therein enhancing network utilization efficiency.

Science uses the language of math but the technical world largely defaults to English to articulately convey and communicate. There are many battles yet to enter the war to keep digital data flowing seamlessly and effectively. It is the obligation of engineers to ensure that the tremendous amount of digital data in transmission presently, and in the future, is not hindered. Edge computing is a significant development on the way forward that will help engineers win the day. M

Requirements for Edge / Fog Systems

by Charles Byers for Mouser Electronics

Selecting markets and determining requirements are the first steps in getting fog or edge enabled:

Selecting Markets

The first step to a solid requirements base is understanding exactly what functions your Internet of Things (IoT) deployment is going to perform. Functionality can be broken down into a three-level taxonomy:

Understand which IoT vertical markets the system will serve (for example, smart factories, connected transportation systems, or smart cities/buildings/homes).

Identify the specific use cases within those vertical markets. These can include areas like surveillance image-processing, building energy management, or rail safety.

Focus on the use cases that target the revenue-producing features of most interest. These can be things like detecting suspicious packages, optimizing elevator movement, or optimizing collision avoidance.

Determining Requirements

Once the basic verticals, use cases, and applications are identified, the specific technical requirements to satisfy those areas can be defined. Examples could include:

- Capacity (user count, storage size, application load)
- Performance (latency, throughput, jitter)
- Reliability (availability, mean time to failure (MTTF), mean time to repair (MTTR), planned outages)
- Security (privacy, authentication, authorization, cryptoprocessing, key management)
- Bandwidth (internode link bandwidth, memory bandwidth, backhaul)
- Management (configuration, orchestration, monitoring, updates)
- Scalability (growing in the above dimensions, modularity)
- Interoperability (working with other suppliers' hardware/software, standards compliance)
- Energy efficiency (average energy use, peak demand, sleep modes)
- Environmental (humidity, "Ingress Protection" rating (IPxx), temperature range, vibration)
- Cost (purchase cost, installation cost, ongoing operational costs, total lifecycle cost)

Once the basic requirements of an edge/fog system are well understood, the design or sourcing of its elements can begin. These requirements aren't static but evolve over time as the vertical/use case/application mix changes and as new technology becomes available.

Network Design: Intermediate Interface Nodes for Critical IoT Network Apps

by Charles Byers for Mouser Electronics

For many critical IoT network applications, relying solely on the cloud for computation, networking, and storage isn't an adequate approach. Instead, intermediate interface nodes that provide computation, networking, and storage between the cloud and the sensors/actuators/IoT are increasingly necessary.

In many critical Internet of Things (IoT) network applications, relying solely on the cloud for computation, networking, and storage isn't an adequate approach. Cloud-centric architectures may not fully meet capacity, performance, reliability, and security requirements (see the Requirements sidebar for more considerations). Instead, intermediate levels of computation, networking, and storage between the cloud and the sensors/actuators/IoT are what is often necessary. Edge, fog, or mist computing are often what these levels are known as. This article focuses on designing and deploying advanced, distributed models for IoT computing, networks, and storage.

Understanding IoT Network Hierarchy

Figure 1 shows a hierarchical deployment of an IoT network. In many IoT applications, all levels of the hierarchy cooperate to provide critical services. Let's look at the levels and their functions individually:

Cloud

The cloud is usually the cheapest and most scalable place to provide computation, networking, and storage for IoT applications, which is evident by the servers, storage engines, and routers contained within it. Unfortunately, the cloud has drawbacks such as high latency, high backhaul-bandwidth costs, reduced reliability, and security issues.

Fog

Fog is a set of cloud-like resources, but "closer to the ground." Fog is often arranged in layers, with more sophisticated functions higher in the hierarchy and lower latency functions lower. For example, higher layer fog nodes may reside in regional or neighborhood settings for smart cities and at the plant or assembly line level for smart factories. Lower layer fog nodes may be located at the street corner level, building level, manufacturing cell level, or individual machine level. Some fog nodes may be mobile, like those found in smart transportation systems.

Edge

Edge computing is usually the lowest layer of the hierarchy, right before the IoT's "things." The edge often includes gateway functions that a system uses to convert between the communications protocols that the "things" use and the formats that the "higher network" uses. Edge nodes may contain low-level control algorithms and modest storage arrays.

IoT Thing / Endpoint

The lowest layer of the IoT hierarchy includes the "things"—that is, the sensors, actuators, displays, and intelligent endpoints the IoT network serves. Some of these devices are very cheap and dumb, while others have built-in intelligence, sometimes approaching that of a smartphone. They connect to other layers of the hierarchy (often through various wireless-link technologies) and rely on the edge, fog, and cloud to perform data analysis, provide storage functions, and run real-time control loops.

Configuring Modular Network Element Hardware and Software

Next, let's explore some of the elements and design tradeoffs necessary during the design and installation of a fog/edge network. These tradeoffs span hardware and software, so let's look at both:

[CONT'D ON NEXT PAGE]

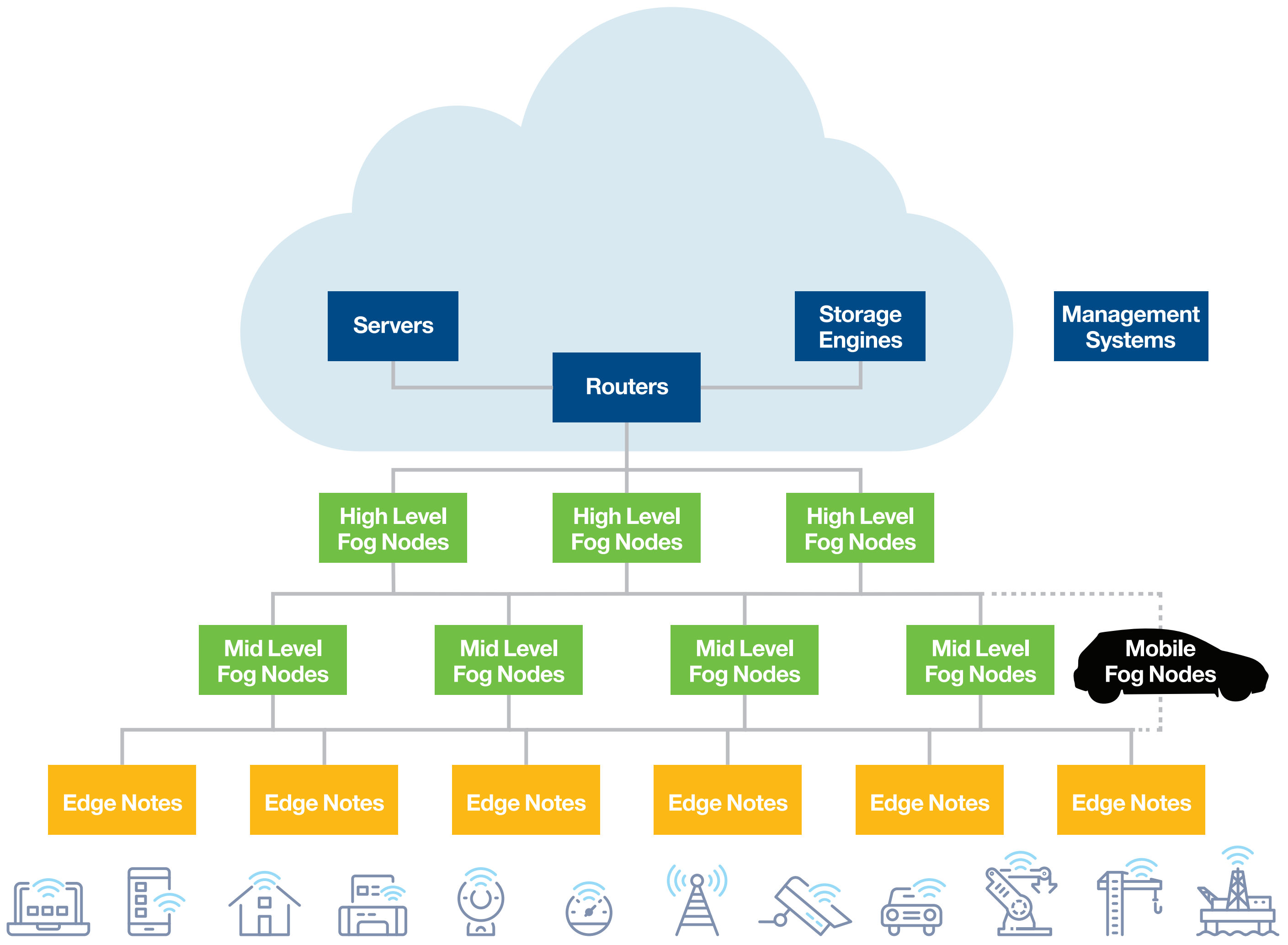


Figure 1: The hierarchical deployment of an IoT network provides different levels of functionality. (Source: Author)

Modular Hardware

Figure 2 describes some of the modular hardware elements that can be valuable in fog/edge nodes. In lower level fog nodes or edge nodes (where power, mechanical size, or cost are very important), a subset of these elements may be soldered directly on a printed circuit board (PCB), with very little modularity or upgrade path. For higher level fog nodes, all the boxes on the figure may be field replaceable modules, using something like a stackable or blade/backplane mechanical design to facilitate equipping the exact modules that complement an application's needs and easily upgrading individual modules in the future. Here are some additional details about these modules:

Processors

Traditional Reduced Instruction Set Computer (RISC) and Complex Instruction Set Computer (CISC) central processing units (CPUs) including x86, Arm, and others are important components of edge/fog processing. They run algorithms that accept, process, and send sensor readings, either towards the cloud or back towards actuators that act based on those calculations. They typically run algorithms that are programmed with conventional software techniques and are often characterized by a need for high single thread performance.

Accelerators

Accelerators are specialized computational elements, designed to run certain workloads much faster or more efficiently than traditional CPUs. One example is a field-programmable gate array (FPGA), where gates are configured to provide custom data paths. Graphics processing units (GPUs) are another example, where the specialized, highly parallel processors designed for graphics rendering can be repurposed to perform IoT application functions. Finally, tensor processing units (TPUs) are an emerging class of accelerators

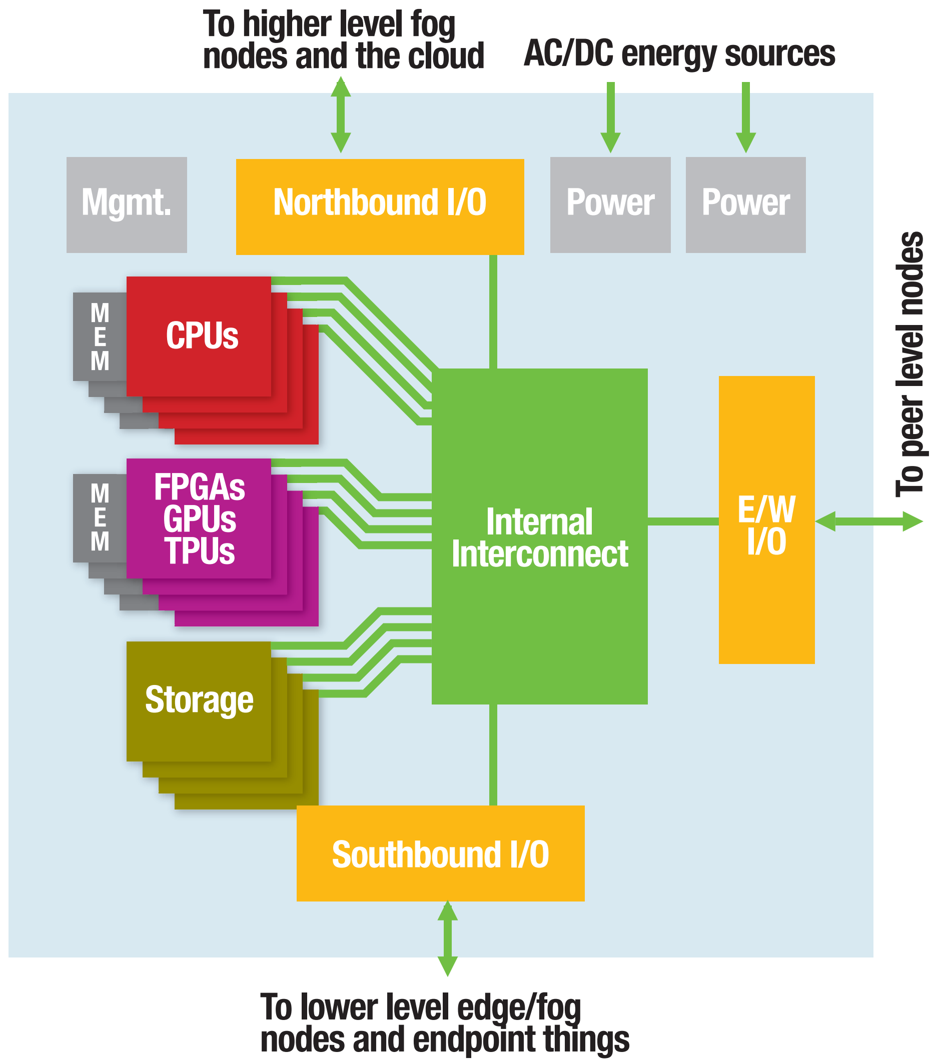


Figure 2: Modular hardware elements can be valuable in fog/edge nodes. (Source: Author)

designed to optimize the execution of artificial intelligence or machine learning functions. Accelerators often have a higher performance per dollar or per Watt than traditional CPUs, especially on computational problems that can run as many parallel threads.

Memory

Edge/fog nodes will require significant memory, both for storing sophisticated platform and application code and for providing intermediate storage for the torrents of data that many IoT applications will produce. The processors and accelerators will act on the data in this memory, filtering and compressing it before sending it to storage or up the edge/fog/cloud hierarchy. Tens of gigabytes or more may be required at each processor.

Storage

Often, local storage is necessary on fog/edge nodes. This is typically an array of flash memory that is used to permanently store the large data structures used in IoT applications. There are tradeoffs in choosing the design of this storage, between rotating disks (which are cheap but may not be too reliable) and solid-state storage (which may have write cycle limits, which may call for using single-level cell (SLC) or other high endurance technologies). Terabyte capacities will be a requirement on many nodes.

Input / Output

Input/output (I/O) systems for edge and fog nodes are divided into two broad categories: Wired and wireless. In general, wired I/O is preferred wherever it is feasible, both because it usually has much better performance and reliability and also because spectrum is scarce and expensive if licensed. It is best to reserve wireless I/O for those applications where it is essential, such as to connect with mobile or portable IoT elements, for example. Wired I/O can be further divided into copper facilities (such as Ethernet, Power over Ethernet (PoE), Controller Area Network (CAN) bus, etc.) and fiber optical links. Wireless I/O is great for mobility and quick link configurations. Wireless I/O

[CONT'D ON NEXT PAGE]

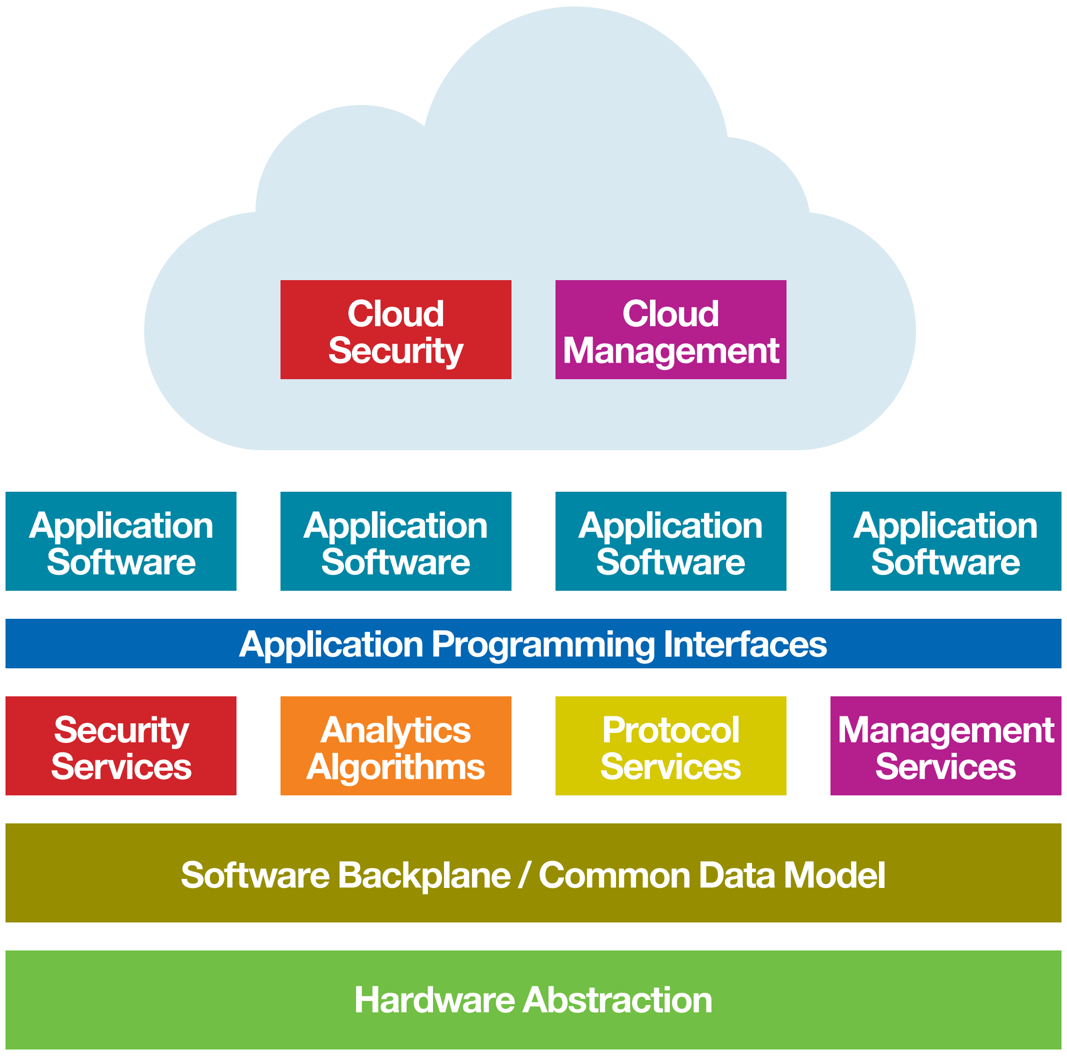


Figure 3: Sophisticated software complements the hardware in edge/fog systems. (Source: Author)

can be divided into two categories: Licensed spectrum (3G/4G/5G cellular networks, satellite links, etc.) and unlicensed spectrum (Wi-Fi, Bluetooth, and LoRa). As shown on the figure, there may be different I/O interfaces for the northbound links, which accommodate higher level fog nodes and the cloud; southbound links, which accommodate the lower fog nodes, edge nodes, and things; and east-west links, for the peer-to-peer communications among nodes.

Support Elements
Edge/fog nodes need several support elements to complete their design. The first one is a power system, which can accept either alternating current (AC) grid-energy sources or direct current (DC) sources (from batteries or renewables). The power system safely and reliably distributes energy to the remaining elements and can be redundant. There is also a management subsystem. It is responsible for the configuration, control, and monitoring of the elements. Finally, there are the mechanical elements of the backplane, card cage, chassis, and cooling system. Cooling is often a serious challenge, especially where high-power processors and accelerators are used, if fans aren't permitted, or if the node needs to be environmentally hardened (extended temperatures and IP65 or better levels of environmental capabilities are often required for outdoor or industrial nodes).

Modular Software

As shown in **Figure 3**, there are sophisticated software elements in edge/fog systems to complement the hardware shown in **Figure 2**. There are literally dozens of protocol-stack diagrams produced by organizations like the OpenFog Consortium, ETSI's Multi-access Edge Computing, Industrial Internet Consortium, and others. This section will try to provide a generic view of the most essential software elements for edge/fog platforms and applications:

Hardware Abstraction
The hardware abstraction layer basically hides the details of the hardware from the software. If

implemented correctly, the hardware abstraction layer will allow us to change elements of the hardware without requiring any modifications to the software. Similarly, the software can change without any impacts on the hardware.

Software Backplane / Data Abstraction
It is necessary to provide a platform infrastructure to bind all the software modules together into a cohesive system. There are two models by which to do this: By the "software backplane" and "system data" models. The software backplane (rather like a hardware backplane) provides several "slots" with standard interfaces into which one plugs various modular software components. Software modules don't have to interact seamlessly with all the other modules, they just have to connect through the software backplane that manages all the interfaces for the platform. Similarly, there is a system data model for the platform: It converts all the data representations, which are in use by all the platform applications, into a common system format; provides proper security; and makes the data available for all the modules.

Security Services
Security is perhaps the biggest challenge for edge/fog software. Two sets of security capabilities are shown in the figure. One is resident in the edge/fog node, and one is virtualized in the cloud that manages the security for the network. The node level security functions include key management, crypto-processing, authentication, privacy, and many other node-level security functions. The cloud-level security process is responsible for policy, identity, key generation, and many other network-level security functions.

Management Services
Management is also an important challenge for IoT networks. The model has two sets of management capabilities. One is in the edge/fog node, and another is virtualized in the cloud. Node-level management performs functions including configuration, monitoring, fault tolerance, orchestration, load balancing, and a host of additional node-level management functions. The cloud-level management system performs network-level functions, like network orchestration, software updates, alarm reporting, and administrative accesses.

Protocol Services
The software platform includes services that are available for use by the rest of the software modules. One example is protocol stacks, supporting the communications between software modules in a node and also between nodes in the network.

Analytics Algorithms
An important function of edge/fog computing is distributed analytics. Some of the analytics algorithms may be supported as platform services for use by all modules. Examples include packet analysis, machine vision, radio frequency (RF) signal intelligence, etc. Also, artificial intelligence and machine learning algorithms will appear here.

Application Programming Interfaces
There is an application programming interface (API) between the edge/fog platform software and the application software that rides upon it. This API allows the simple creation, deployment, and updates of the hardware and software modules below it as well as the applications' programs above it. Ideally, this API should be standardized, but the standards community is still working to converge on a single API for this purpose.

Application Software
This is the reason all this software and hardware exists—to enable network applications that provide real business value. This software may be written by several different groups, including the supplier of the hardware or software platforms, data network operators, system integrators, node owners, end users, and various domain experts. Eventually, there will be an open marketplace for standard applications software that can run on any fog or edge node—similar to the iTunes® or Google Play™ marketplaces. Container models using technologies such as Docker or Kubernetes are often in use for the application's code. Four instances of application software are shown on the figure, of which four different tenants can own on the same edge/fog processor.

Putting It All Together

A concrete example should demonstrate how edge/fog systems fit together. Consider a multicamera, multisensor physical security system for monitoring an installation like an airport, casino, campus, or secure industrial facility. There are requirements for capacity (which include hundreds of cameras and sensors), performance (which is low latency), mission-critical reliability, and excellent security.

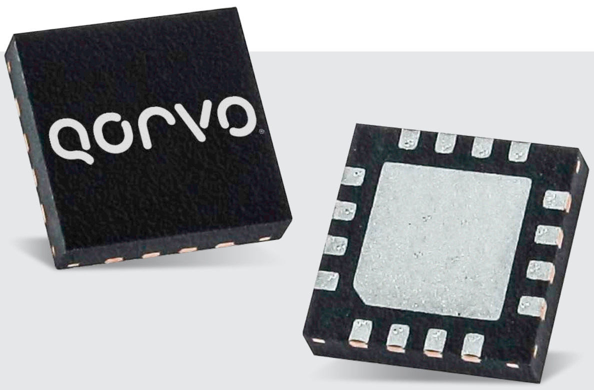
Because of latency and bandwidth constraints, we can't backhaul all the video streams to the cloud, so we must run analytics on a hierarchy of local fog/edge nodes. The video analytics and sensor fusion tasks can be split across levels of the network. For example, this split can occur with

[CONT'D ON NEXT PAGE]

qorvo®

Wi-Fi Front End Modules

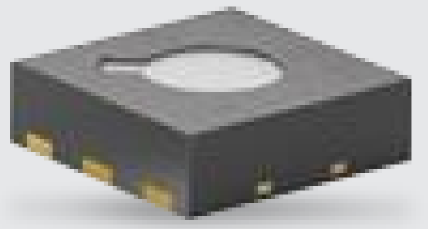
mouser.com/qorvo-wifi-fem



SENSIRION
THE SENSOR COMPANY

Multi-Pixel Gas Sensor SGP

mouser.com/sensirion-sgp-gas-sensor



low-level edge nodes performing feature extraction and correlation tasks, while high-level fog nodes perform object recognition and threat assessment functions. Segregating functions in this way minimizes the network bandwidth and simplifies the correlation between sensor readings and images across multiple sensors.

Likewise, the modular hardware of the edge/fog nodes ensures that an optimal mix of processors, accelerators, storage, and I/O interfaces are available on each network node. The platform software ensures that all nodes are secure, correctly managed, and include the essential platform functions.

In general, the application software (also running on the local nodes) can be written by diverse sets of experts and can perform many smart building and surveillance-related functions. However, only the highest layer of reporting and policy management can touch the cloud. This is a very versatile and efficient deployment model for complex applications.

Conclusions & Next Steps

As you can see, there is a rich canvas to paint upon with the emergence of edge/fog capabilities for IoT networks. Once the verticals, use cases, and applications are chosen and requirements are articulated, you can then assemble various modular hardware and software components into node-level and network-level solutions. The next step is to deploy those systems in real-world IoT network applications (please see the sidebar for some additional thoughts on the deployment process).

Considering how important IoT is becoming in our lives, and the important challenges to meet in making it work securely, reliably, and efficiently at a high scale, the edge/fog techniques described in this article will soon be extremely valuable. M

Deploying Edge / Fog Systems

by Charles Byers for Mouser Electronics

Once the application refinement and requirements are complete, and the hardware and software have been assembled, it is time to deploy a network of edge/fog nodes. There are several steps involved in this deployment.

Design

A network of fog/edge nodes are designed and built according to the principles of the preceeding article.

Installation

The nodes are taken to their installation sites and physically installed. This may involve securely bolting the chassis permanently in place, connecting network cables or positioning antennas, providing reliable power sources, and establishing cooling facilities.

Configuration

Every node in the network must be configured to establish its network addresses, load its security keys, install the correct versions of its platform and application software, and download any initial data, tables, content, media files, or user information that may be required. Automation of the repetitive tasks is particularly important at this stage.

Testing

Before bringing a node or network into service, it should be rigorously tested. These tests could include mainline functional tests, performance tests, stress tests under abnormal loads, or verifications that the fault recovery function works as intended.

Commissioning

The network is given live traffic for the first time. Generally, it's good to go slow here, first with small numbers of friendly users and the less critical Internet of Things (IoT) network functions and then gradually growing to full capacity with the full complement of its functions.

Long-term operations

Once the network is handling a full traffic load with its full complement of functions, it must be carefully monitored for abnormal conditions like power problems, environmental concerns, security threats, overloads, etc. Also, the system network requires nearly continuous updating including platform and application software revisions, probably on at least a weekly basis. Modular hardware can also be updated several times during the life of a node.

These steps allow the smooth deployment of edge/fog capabilities, enabling them to be key functional elements of advanced IoT systems.

How AI at the Edge Will Change Engineering

by M. Tim Jones for Mouser Electronics

Machine learning at the edge will drive changes in processor architectures, storage technologies, and communication interfaces and protocols. While the edge is commonly driven by lower-powered technologies, edge devices will embody a spectrum of computing and storage capabilities for machine learning.

Early computing systems consisted of large and expensive mainframes that epitomized centralization. Microprocessors in the 1970s brought a digital revolution in computing that decentralized these resources and made computers accessible everywhere. But in the last decade, centralization returned with the advent of cloud computing, which used economies of scale to decrease the cost of computing and storage on demand.

The pendulum is now swinging once again to decentralization, partly due to the Internet of Things (IoT) and intelligent devices, but in a way that optimizes where computation and storage are allocated. This concept called fog computing considers the quality-of-service and where it is best to apply machine learning using data at the edge.

An example of this concept is a smartphone's voice assistant. When you ask a question, the audio is recorded and sent into the cloud for voice recognition and language understanding. The request is processed, and the results are returned to you smartphone: Your voice is captured at the edge and then processed in the cloud. Contrast this with a self-driving vehicle that can't process its images in the cloud and must instead process and understand these images locally to make fast decisions. Fog computing is about processing data in the right place.

Changes for Machine Learning at the Edge

With the rise of machine learning at the edge, the hardware and software ecosystem that supports data collections, processing, storage, and the machine learning algorithms is adapting to this changing environment. Let's explore how this ecosystem is adapting to machine learning at the edge.

Processor Architectures

Machine learning can be computationally intensive, but this depends on the algorithms in use. Deep learning algorithms can require extreme performance per watt, but simpler statistical algorithms for classification or prediction can operate on tiny microcontrollers whose power consumption consumes something on the order of several tens of milliwatts. But processor vendors have taken note of the boom in IoT and edge machine learning and are now adapting to this challenge. Arm announced a new processor architecture called Project Trillium that includes two processors focused on high-performance machine learning and object detection for 60Hz video. Intel has also announced a Xeon processor designed for edge applications that support up to 18 cores.

Processor vendors have also seen instruction set opportunities for machine learning, optimizing and creating new instructions focused on neural networks (the basis for deep learning algorithms). These instructions combine, multiply, and add operations that make up the building block for inter-neuron communication.

Graphics processing units (GPUs) continue to be the workhorse for deep learning applications, whose architectures are built for highly-parallelized, complex image processing, but new architectures are also being defined to accelerate neural networks. Google's Tensor Processing Unit (TPU) is a custom application-specific integrated circuit (ASIC) whose instruction set was designed from the ground up for neural network tasks. Google has even tailored the TPU for use at the edge with a focus on a smaller physical footprint and lower power consumption.

[CONT'D ON PAGE 46]

```
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif _operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

    #selection at the end -add back the
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modi
print("Selected" + str(modifier_ob)) #
```

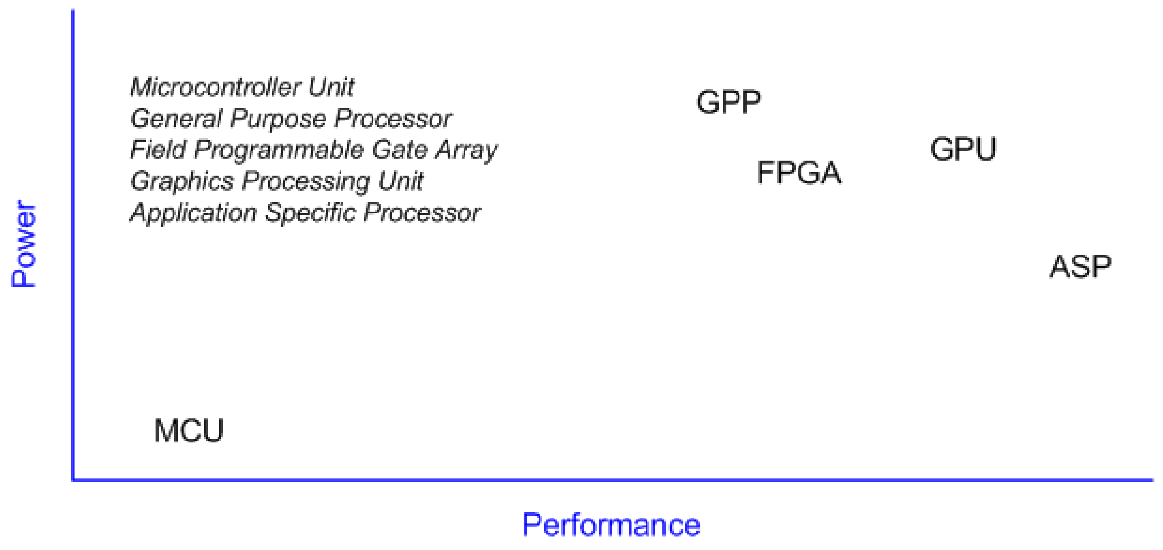


Figure 1: This image depicts the power versus performance of various processor types. (Source: Author)

Figure 1 illustrates the evolution of processors from low-performance microcontrollers to high-performance application-specific processors. Edge machine learning drives higher performance levels and lowers power consumption in application-specific architectures tailored for machine learning applications.

Storage Architectures

Arriving at a classification in a deep learning network involves many layers of multiply-add operations over elements of the image that are transformed through weights in the layers of networks. A reasonably sized deep learning network can include 30 million weight parameters with 20 million activations. Assuming a typical 32-bit floating-point value, that's a storage requirement of 200MB just for the network weights and operating data (excluding the application, input image data, etc.). The image data and operating data are transient and change over time, where the weights are static (but may change with the network's periodical updates). This pattern creates a requirement for the types of data: Persistent versus non-persistent. The network activations are performed in non-persistent memory, where the application and weights are initially stored in persistent memory. But these categorizations are being blurred by new technologies. NAND flash is replacing traditional rotating media in many applications, particularly in embedded applications that make up edge computing.

But traditional hard drives and solid state drives are not the only options, and new technologies are driving new applications. One such technology is called phase change memory (PCM), which is persistent like NAND, but rather than operating on storage interfaces, PCM with its higher performance can expose a memory interface (**Table 1**). This new type of non-volatile memory is called a storage-class memory. And while not as fast as dynamic random-access memory (DRAM) (particularly for writes), PCM can operate with memory semantics, making it ideal for data like weights, which may not change often but are read often at a high-performance level.

Solid-state storage can also be preferential over hard disk drives (HDD), when considering the environment in which the edge device will exist. HDDs, with their moving parts, are highly susceptible to shock and vibration, where solid state devices escape this limitation.

These types of memories and storage technologies provide the means to optimize your machine learning dataflow, taking advantage of the read and write characteristics of each type. This means getting data closer to the processor, where it can be more efficiently processed to make faster decisions.

Communication Technologies

In a traditional edge solution, data is collected from its local environment and then communicated into the cloud for analysis. Pushing machine learning into the edge imbues the device with autonomy and the ability to act in isolation. This can be advantageous in several scenarios that involve limited connectivity, limited bandwidth, or a mismatch between bandwidths and the required amount of data to communicate for a given set of interactivity metrics.

But even with highly autonomous edge solutions, communication is still required for management and monitoring, auditing, and other device-specific requirements. Edge computing and the IoT are driving technologies such as 5G. Fifth-generation wireless, or 5G, is the latest cellular technology that will support speeds of 1Gbps.

In cases where data is not exposed to the cloud for analysis, a wearable medical device that uses a recurrent neural network to detect atrial fibrillation could use 5G to quickly transfer data for further analysis after a local prediction. Hybrid solutions are also useful where portions of machine learning are performed at the edge and then completed in the cloud. Particularly in the context of neural networks, input data, such as images, can be reduced by early layers of a neural network resulting in features (e.g., higher-level abstractions of the input data) that can be communicated to the cloud for further processing.

The growth of edge computing will quickly capitalize on the new bandwidth created by 5G to create new markets. Other technologies such as Narrowband IoT (NB-IoT) or low-power wide-area networks (LPWANs) will fill in the gaps for speed, range, and low-power requirements.

Machine Learning Algorithms at the Edge

It's not just hardware technologies that are evolving at the edge for machine learning, machine learning is adapting as well to this environment. While deep learning algorithms tend to focus on high-performance hardware, the algorithms themselves have been tailored to resource-constrained environments such as embedded systems at the edge. TensorFlow Lite is an example of scaling the popular TensorFlow environment as a lightweight solution for mobile and embedded systems. It optimizes deep learning through a variety of techniques such as fixed-point math but can also leverage the Android neural networks application programming interface (API) for hardware acceleration.

Scaling down even further, the uTensor project focuses on bringing machine learning to low-power and low-cost microcontroller units (MCUs). The uTensor project has demonstrated a three-layer perceptron neural network that was trained in 256KB for the Modified National Institute of Standards and Technology (MNIST) handwritten digits database. The developers have included support for Arm's Cortex machine-learning APIs and are working on convolutional neural network (CNN) and recurrent neural network (RNN) implementations.

Deep learning architectures using CNNs are also evolving for edge-based embedded systems. The SqueezeNet architecture combines point-wise filters and late down-sampling to maintain a large feature map. But the nature of SqueezeNet's layers involves squeezing the feature map size and later expanding it using a smaller number of layers. This CNN architecture maintains useful levels of accuracy (compared to AlexNet), but in a model that requires 500 times fewer parameters and uses about 500KB of memory along with deep compression. Google also released MobileNets as an open-source implementation for on-device neural network vision applications. MobileNets includes an interesting property through two local hyper-parameters that support an efficient trade-off between speed and accuracy.

Small deep neural networks are being applied to a range of practical problems in object classification and detection methods, with acceptable accuracy in platforms like smartphones and similar hardware platforms.

Convergence at the Edge

As the demand grows for intelligence at the edge, technology vendors and machine learning engineers will find new ways to satisfy these requirements. Through new technologies that evolve to support today's machine learning algorithms or through new approaches to machine learning on ever-shrinking platforms, fog computing will merge algorithms and data for tomorrow's applications. M

STORAGE	PERFORMANCE	COST	PERSISTANCE	ENDURANCE	INTERFACE
HDD	+	+	*Non-volatile*	++++	*Storage*
NAND	++	++	*Non-volatile*	++	*Storage*
PCM	+++	++++	*Non-volatile*	+++	*Memory*
DRAM	++++	+++	*Volatile*	++++	*Memory*

Table 1: PCM is a new type of non-volatile memory that is ideal for data that doesn't change often but that's read at a high-performance level. (Source: Author)

Device Security in a Larger Edge-Cloud Model

by Jeff Fellinge for Mouser Electronics

Embedded devices require security care like all other servers running full-featured operating systems. Reduce the risk of introducing new security vulnerabilities and protect the entire stack by using effective security controls on your devices, networks, the edge, and cloud systems.

Embedded devices are an important part of edge services. They locally collect and process data and instructions from a variety of sensors and equipment. Thus, they require similar security care and attention as that of any other server that is running a full-featured operating system. Lower costs and increased performance of embedded devices are making it easier than ever for distributed systems, equipment, and machinery to connect through Internet-connected cloud services. Whether for use in homes and small businesses or to manage more critical industrial-control systems, the ability to connect seems endless through the use of these devices.

Unfortunately, for these same reasons, the seemingly ubiquitous and often well-connected systems also attract attackers. A security compromise of even one embedded device may not only give an attacker access to view or manipulate sensitive data that the device has stored or processed but also potentially gives an attacker a foothold into a trusted network. Thankfully, the security of these devices is steadily improving through the new release of more prescriptive security guidance and continuous advances in the devices' security capabilities.

Systems engineers must understand the security capabilities of an embedded device, how the device operates, and what data it collects and processes. Only then can the engineers appropriately select, design, and deploy the right equipment into an environment in a way that remains consistent with the broader security goals and objectives of that specific environment. This article explores four tips for device security within a larger edge-cloud model:

- Keep the bigger picture in mind
- Know your security requirements
- Meet security requirements through effective controls
- Always pay attention to the fundamentals

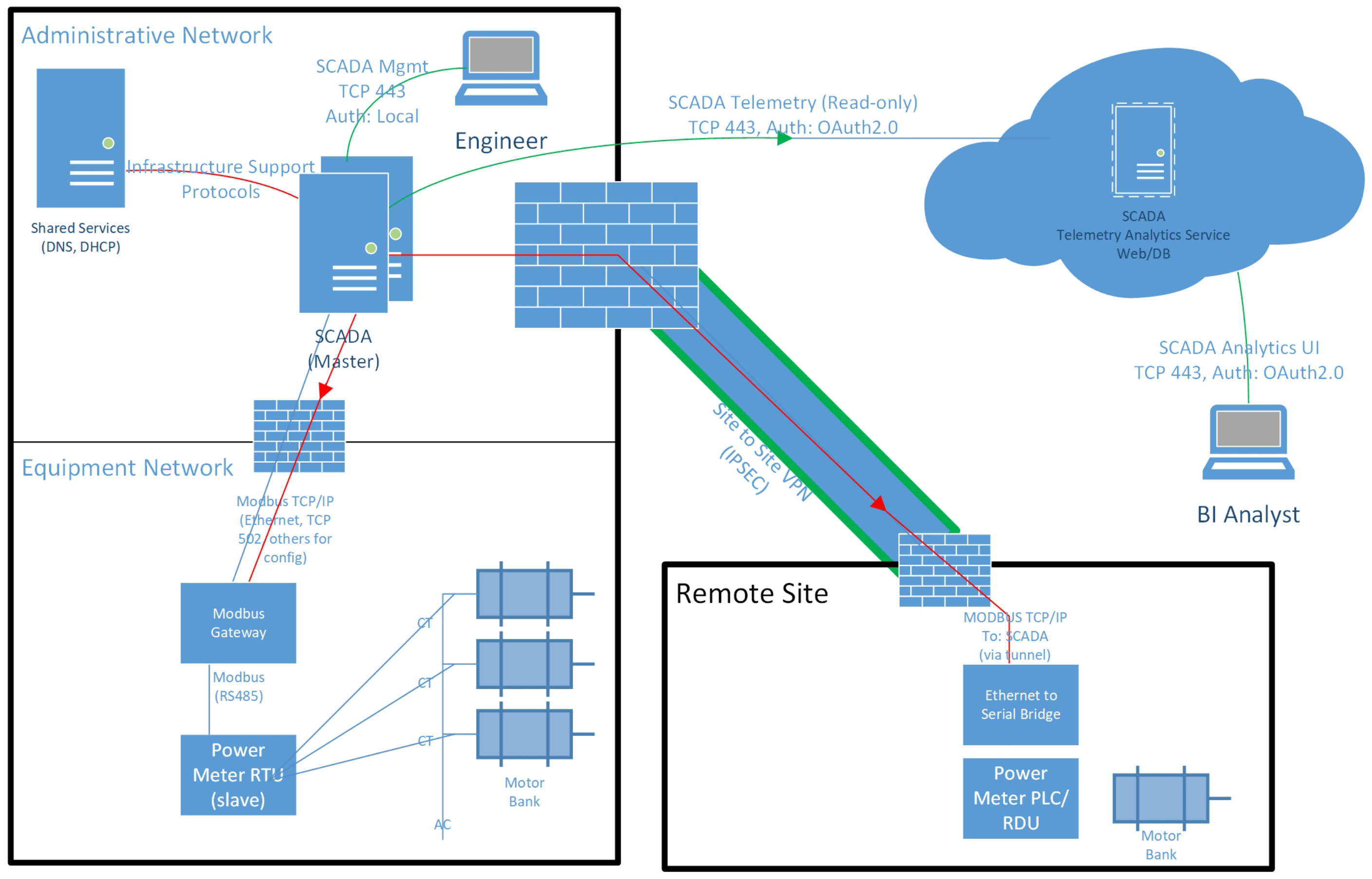
Keep the Bigger Picture in Mind

When designing or implementing the security capabilities of a component or device, in your review, include the broader system to which it connects. This helps to avoid introducing new security vulnerabilities to or unknowingly inheriting risks from the larger system. For example, will the new device control critical machinery, either directly or through data collections via telemetry? Will it bridge two networks, such as connecting a private or trusted network to the Internet? How and when will people interact with the device, and how will individual components link to create larger systems?

Answers to questions like these will help guide the design to the right security controls to protect not only the component but the broader system as well. Use this knowledge to leverage the security capabilities of nearby applications and platforms. For example, cloud services such as Microsoft Azure and Amazon Web Services (AWS) provide application programming interfaces (APIs) that make it easier to tap into the cloud provider's security controls to help secure connected devices and systems.

Document what you are learning as you go including the goals and objectives of the new component or system, use cases and assumptions that define the system, and the overall architecture. One particularly useful tool for analyzing the network security of a system is the data flow diagram (DFD), which shows paths of communication between devices and systems. The DFD, like the simple one shown in **Figure 1**, also includes useful metadata about the communication flow, such as

[CONT'D ON PAGE 50]



encrypted 🟩
unencrypted 🟥

Figure 1: This data flow diagram shows the security-related networking characteristics of a simple embedded device and system configuration. (Source: Author)

what network traffic encryptions exist, what authentications are at work, what protocols are in use, and where devices logically reside on the network. DFDs are a critical part of threat modeling to identify where gaps may exist between the design and the security objectives.

Keeping the bigger picture in mind is important for everyone involved in the project—from the component engineer that designs the hardware, the software engineer that writes the firmware, the systems engineer that deploys the components, or the DevOps team that manages ongoing maintenance. While the knowledge level will vary by task and responsibility, each of these roles plays an important part in making a system both fully functional and appropriately secure. Be sure everyone knows their role in meeting the security requirements of the project. Many published security requirements are written with specific roles in mind, which makes it easier to find and assign the right people to own, manage, and implement controls that meet these requirements.

Know Your Security Requirements

Your project's security requirements may be defined by your corporate security policy, customer commitments, or the vertical your business serves. For example, if your system is exposed to credit card data or personal health information, you may be required to meet very specific security objectives and regularly demonstrate compliance. Different standards organizations, like the National Institute of Standards and Technology (NIST) and Underwriters Laboratory (UL), publish a variety of security requirements and guidance for network-connected products and cloud applications and platforms. While the number of individual requirements may seem daunting, fortunately, these requirements often overlap, and your security controls that meet one set of requirements may often satisfy others as well.

Meet Security Requirements through Effective Controls

Security is made up of repeatable, effective controls. When designing a component for use in an edge or cloud scenario, a security requirement is only fully met when the controls that map to this requirement operates effectively across the entire stack—that is, beginning from lower level components and devices up through the edge systems and into the cloud. A control failure at any point could break this chain and result in a broader security breach or system failure.

Take, for example, the security requirements surrounding authentication and authorization, two key concepts of identity management. Authentication (proving who you are) and authorization (what the system allows you to do) play a critical role in access security not only for a device or system but also for services further up the stack, like, for example, the network the device connects to and the edge or cloud service that will process and store data that the device collects.

Leaked or stolen credentials are a leading cause of system compromise. Often through phishing, an attacker will attempt to steal authentication credentials, such as usernames and passwords, then attempt to impersonate the victim on websites or access their email. Sophisticated attackers will use stolen credentials to access corporate networks and cloud subscriptions to steal valuable data or disrupt the system's performance. Once an attacker has a foothold in a remote system, they may try to pivot to a more valuable target. Overly broad access from a poor authorization scheme makes pivoting much easier for an attacker.

One way to reduce this risk is through role-based access control and identity isolation. For example, restrict identities for day-to-day activities, like viewing websites and email, from use as the same identities to access production networks or cloud subscriptions. In this manner, one careless misstep by a user on their laptop should not put your larger system at risk. Also, while attackers will use stolen credentials and vulnerabilities to jump from one system to another (lateral movement), they may try to access higher privilege, supervisory functions in a single system (vertical movement). Combat this access by restricting privileged accounts, such as the root or administrator, from remotely logging into your device or system, by logging all superuser activity, and by regularly reviewing the logs for questionable behavior.

In an edge and cloud model, it is important to keep in mind that other people (not you) will manage the security controls of the systems that your device must connect to further up the stack. The accountability and responsibility of the security controls across this stacking model vary according to whether the systems are on the premise, on the cloud, or on a hybrid of both. You may find that some of these security controls that others manage do not meet your security requirements. In these cases, look for and implement additional controls to compensate for these gaps.

Generally, the further up the stack the larger the blast radius—or number of systems that have the potential for disruption by an attack. For example, an attacker compromising one device on the network might not be able to cause as much damage as they would if they breached the network equipment serving multiple devices. Attackers know this and will focus their efforts to discover and breach targets with broader impact.

Keep this in mind as you are creating your architecture. Reflect your decisions in your DFD as you establish the threat model for your design. Here are some additional architectural principles to consider when designing your controls:

Understand Potential Consequences of Dependency

Know what authentication and authorization services are available for your project. Understand that taking a dependency on another service might have consequences, such as opening a firewall to allow communication with an upstream identity provider. Some Internet of Things (IoT) and embedded devices or services do not support modern authentication frameworks and require local accounts. In these cases, and where the device performs a critical function, additional controls must be put into place to further protect the device.

Consider the Impact and Potential Blast Area

Consider the impact and potential blast area of a compromised account—e.g., could an attacker use this identity to see some or all sensitive and private data? Is an attacker confined to the device that he or she breached, or could this person use this system or identity to reach other devices? For example, a cloud subscription compromise, by way of a leaked or stolen identity, could enable an attacker to

[CONT'D ON NEXT PAGE]

reconfigure an entire cloud service or network, change code, or steal data from any cloud applications that the cloud account manages.

Enable Multi-Factor Authentication

For highly sensitive accounts, be sure to enable the elements of a multi-factor authentication process to reduce the risk of password theft. Such elements include:

Automated Controls
Use automated controls where possible to reduce long-term costs and avoid manual mistakes. For example, a centralized or federated identity solution is generally less error-prone than managing distributed, individual, local accounts for each user on every device.

Isolated Identities
Use isolated identities based on function. For example, configure industrial-control system equipment with a different identity-management system than what operators use for email and web access.

User and Access Provisioning
Decide how you want to execute user and access provisions. Consider role-based access control (RBAC) to limit overly broad access. For example, be sure you know where you must use access elevations, via a root/sudo or another superuser, instead of an account with lower privileges. Regularly audit group memberships (e.g., the Administrators group in Active Directory) and look for nested groups that might hide the true number of people with access privileges.

More Security Controls

Authentication and authorization are just two examples of security controls for managing identities. There are many controls similar to these across other security domains, including:

- Surface area protection and network access control lists, like network and host-based firewalls
- Encryption for data in transit and data at rest
- Activity logging for unexpected behavior, deviation alerts
- Vulnerability and security update management
- Configuration and change management of the devices themselves

Know Your Cloud Service's Security

Know how your cloud service secures data access. Before you configure your edge devices and upload sensitive data, know how your cloud service secures access and consider the same security requirements that you use for your device and edge solutions. For example, how will your device authenticate an identity for cloud service? Are these credentials different from those of the operators that log into the cloud for maintenance?

Always Pay Attention to the Fundamentals

Even when your use case may not warrant sophisticated security controls, it is always important to maintain strong fundamentals to ensure that when action is required, or a breach occurs, your response is quick and complete:

- Keep a complete and accurate inventory of every device connected to your network. You must know what you have before you can begin to secure it.
- Record and manage the right metadata—your device's make, model, location, Internet Protocol (IP) address, supported protocols, and operating system (OS) versions/firmware versions. This is especially important when considering the proliferation of different types of systems that are running minimal but very capable firmware, including Arduino, Raspberry Pi, and even proprietary programmable logic controllers (PLCs).
- Follow your manufacturers' recommendations for updating any system firmware and subscribe to published security bulletins.

Where you may have less capable systems or devices performing in more important or critical roles, always remember to identify and implement compensatory controls to help isolate these devices.

Security Requirements for Devices

Security requirements define the necessary controls to meet your security and compliance obligations. These requirements may be set internally by a corporate security policy or externally by independent standards organizations. Specific security requirements may also be set by the country, region, or jurisdiction in which the business operates. Think of all the security requirements as an extension of the business objectives and commitments made to customers. When designing a new device or product, it is important to understand all the sources of applicable security requirements.

Once you have identified your security requirements, you must then define the controls to meet these requirements. Many companies rely on internal and external assessments, and audits ensure these controls operate effectively. Certifications from compliance audits demonstrate to customers and stakeholders that a company, product, or service meets its stated objectives. And while these processes and tools are often a part of more formal governance, risk, and compliance (GRC) program used by risk managers and security professionals, the methodologies and security content can be very important and useful for systems designers, engineers, and operators alike.

Audits measure the effectiveness of your security controls and ultimately whether your security requirements are met. Audits can be managed either internally or by a third party, as a part of a broader certification process. During an audit, as an engineer, you may be asked to show how a security control operates. For example, you may have to create a list showing all the privileged users on a system or demonstrate how a device's configuration is safeguarded from tampering.

International standards describe these requirements for different types of cybersecurity frameworks. The International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27001 standard provides the requirements for an information security management system (ISMS) that can help organize your security program and ensure that the right systems and processes are in place to protect your business and product assets from identified risks. ISO/IEC 27002 includes examples of different configuration options and can be a great source of help for new adopters.

Industry verticals like healthcare and banking define specific security requirements appropriate for their customers, products, and data. For example, if your device or product processes or stores credit card data, then you may be subject to the Payment Card Industry Data Security Standard (PCI DSS). If you process or store electronic protected health information (ePHI) in the United States (US), then you must meet the security requirements defined in the Health Insurance Portability and Accountability Act (HIPAA) and Health Information Technology for Economic and Clinical Health (HITECH) Act. US government customers may have to satisfy the Federal Risk and Authorization Management Program (FedRAMP) requirements, which identifies security and privacy controls and assessment procedures from the National Institute of Standards and Technology (NIST) Special Publication 800-53 standard. Even if you are not required to meet FedRAMP, NIST SP 800-53 is a terrific (and free) resource to identify best practices and guidance for securing information systems and organizations. The latest version, NIST SP 800-53 Revision 4, defines hundreds of security controls across 18 different families (e.g., access control, identification and authentication, incident response, risk assessment, and system and communication protection).

Additionally, there are many commercially available standards that target more niche markets. The Underwriters Laboratories (UL) 2900-1 standard for safety, software cybersecurity, and network-connectable products defines security requirements specifically for devices and products. The International Society of Automation (ISA) 99 standards also describe a cybersecurity management system for automation and control professionals protecting critical infrastructure.

The amount of available information can be overwhelming. Several standards organizations have created control frameworks that identify overlap between standards and distill these security requirements into best practices and prescriptive guidance that are more easily adaptable to product designs. For example, the Cloud Security Alliance (CSA) publishes the Cloud Controls Matrix (CCM), a spreadsheet that defines security control specifications and architectural relevance across the entire stack—from the physical layer through the data layer. This matrix cross-references cloud-relevant security controls with other security standards to create a more concise list. M

Authors

Charles C. Byers

Charles “Chuck” Byers is a Senior Technical Lead and Platform Architect with Cisco’s Enterprise IoT Group. He works on the architecture and implementation of Fog Computing platforms, media processing systems, and the Internet of Everything. Before joining Cisco Systems, he was a Bell Labs Fellow at Alcatel-Lucent. He has also been a leader in several standards bodies, including serving as a founding member of PICMG’s AdvancedTCA, AdvancedMC, and MicroTCA subcommittees, and currently serves as co-chair of the Architecture Framework Working Group and Technical Committee of the OpenFog Consortium. He holds 77 US patents.

Stephen Evanczuk

Stephen Evanczuk has more than 20 years of experience writing for and about the electronics industry on a wide range of topics including hardware, software, systems, and applications including the IoT. He received his Ph.D. in neuroscience on neuronal networks and worked in the aerospace industry on massively distributed secure systems and algorithm acceleration methods. Currently, when he’s not writing articles on technology and engineering, he’s working on applications of deep learning to recognition and recommendation systems.

Jeff Fellinge

Jeff Fellinge has over 25 years’ experience in a variety of disciplines ranging from Mechanical Engineering to Information Security. Mr. Fellinge has experience reducing risk and improving security control effectiveness at some of the world’s largest datacenters. He enjoys researching and evaluating technologies that improve business and infrastructure security and also owns and operates a small metal fabrication workshop.

Paul Golata

As a Senior Technical Content Specialist at Mouser Electronics, Paul Golata is accountable for contributing to the success in driving the strategic leadership, tactical execution, and overall product line and marketing direction for advanced technology related products. Prior to Mouser Electronics, he served in various Manufacturing, Marketing, and Sales related roles for Hughes Aircraft Company, Melles Griot, Piper Jaffray, Balzers Optics, JDSU, and Arrow Electronics. Mr. Golata holds a BSEET from DeVry Institute of Technology (Chicago, IL); an MBA from Pepperdine University (Malibu, CA); and a MDiv w/BL from Southwestern Baptist Theological Seminary (Fort Worth, TX).

Bin Jiang

Bin Jiang received B.S. and M.S. degrees in communication and information engineering from Tianjin University, Tianjin, China.He is also a visiting scholar in Security and Optimization for Networked Globe Laboratory, Embry-Riddle Aeronautical University. He is the author of more than 30 papers, and he also served as reviewers for several journals. His research interests lie in the Internet of Things, edge computing, big data analytics, and image processing.

M. Tim Jones

M. Tim Jones is a veteran embedded firmware architect with over 30 years of architecture and development experience. Mr. Jones is also the author of several books and many articles across the spectrum of software and firmware development. His engineering background ranges from the development of kernels for geosynchronous spacecraft to embedded systems architecture and protocol development.

Chaofan Ma

Chaofan Ma received a B.S. from the School of Materials Science and Engineering, Tianjin University, Tianjin, China. He is currently pursuing an M.S. degree at the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His research interests lie in edge computing, wireless edge cache, and mathematical optimization for networks.

Jason Shepherd

Ranked in IoT ONE’s list of Top 100 Industrial IoT Influencers of 2018, Jason Shepherd is CTO for IoT and Edge Computing at Dell Technologies. In this role, Mr. Shepherd drives IoT/Edge market and technology strategy, standards, solution planning, and strategic ecosystem development, including building the Dell IoT Solutions Partner Program from scratch, which received the 2017 and 2018 IoT Breakthrough Award for Partner Ecosystem of the Year. Recently, Mr. Shepherd led the creation of the EdgeX Foundry open source project to facilitate interoperability at the IoT edge.

Houbing Song

Houbing Song received a Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in August 2012. In August 2017, he joined the Department of Electrical, Computer, Software, and Systems Engineering at Embry-Riddle Aeronautical University, where he is currently an Assistant Professor and the Director of the Security and Optimization for Networked Globe Laboratory. He serves as an Associate Technical Editor for IEEE *Communications* Magazine. He is the author of more than 100 articles, editor of four books, and a senior member of both IEEE and ACM.

Huifang Xu

Huifang Xu received a B.S. degree from the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. She majored in communication engineering, and she is currently pursuing an M.S. degree at the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. Her research interests lie in edge computing, wireless content caching, and the Internet of Things.

Engineers and Buyers find the leading brands and the widest selection of products in stock at Mouser

ORDER WITH CONFIDENCE | mouser.com

MOUSER ELECTRONICS

Discover • Design • Develop