EPISODE 777

[INTRODUCTION]

[00:00:00] JM: Many devices in our world are not smart; air conditioners, electric guitars, power outlets, factory conveyor belts. It's quite easy to name the devices in our world that aren't smart and yet there are exciting software applications that we could build around these devices. We just need to know how to interface with them programmatically.

We need to be able to know the state of these devices. We need to be able to save that state data and then we need to be able to use that state data to perform actions that change those devices, and to make these devices smart, we can use a micro-controller, which is a small device with a constraint amount of CPU, memory and I/O.

Device data can be sent to the cloud or it can be processed locally, and that data can be used to perform predictive maintenance, or create machine learning models, or create simple dashboards so that human operators can understand the state of their hardware.

Dirk Didascalou is the VP of Internet of Things with Amazon Web Services. Dirk joins today's show to discuss the strategy and the philosophy of the AWS Internet of Things set of tools. We talk about a wide-ranging set of topics, including IoT security, edge deployments and machine learning. I hope you enjoy this episode.

[SPONSOR MESSAGE]

[00:01:29] JM: DigitalOcean is a reliable, easy to use cloud provider. I've used DigitalOcean for years whenever I want to get an application off the ground quickly, and I've always loved the focus on user experience, the great documentation and the simple user interface. More and more people are finding out about DigitalOcean and realizing that DigitalOcean is perfect for their application workloads.

This year, DigitalOcean is making that even easier with new node types. A $15 flexible droplet that can mix and match different configurations of CPU and RAM to get the perfect amount of resources for your application. There are also CPU optimized droplets, perfect for highly active frontend servers or CICD workloads, and running on the cloud can get expensive, which is why DigitalOcean makes it easy to choose the right size instance. The prices on standard instances have gone down too. You can check out all their new deals by going to do.co/sedaily, and as a bonus to our listeners, you will get $100 in credit to use over 60 days. That's a lot of money to experiment with. You can make a hundred dollars go pretty far on DigitalOcean. You can use the credit for hosting, or infrastructure, and that includes load balancers, object storage. DigitalOcean Spaces is a great new product that provides object storage, of course, computation.

Get your free $100 credit at do.co/sedaily, and thanks to DigitalOcean for being a sponsor. The cofounder of DigitalOcean, Moisey Uretsky, was one of the first people I interviewed, and his interview was really inspirational for me. So I've always thought of DigitalOcean as a pretty inspirational company. So thank you, DigitalOcean.

[INTERVIEW]

[00:03:36] JM: Dirk Didascalou, you are the VP of AWS IoT. Welcome to Software Engineering Daily.

[00:03:41] DD: Hi, Jeff. Thanks for being here.

[00:03:43] JM: So you've said that if users were able to know the state of every device that they are working with across their infrastructure, the would be able to reason on top of that state data and they would be able to solve many more problems than we can solve today. What does that mean to know the state of our devices?

[00:04:02] DD: I think it's very simple. It's just to understand more about the realities of your assets. Think about in a factory. Today, if you want to understand whether a conveyor belt is stuck, typically you have to be there and look at it. What if you knew this automatically in a dashboard? If you're at home and want to understand whether your nice little Roomba robot is doing its job, you don't have to be in the room to understand whether it's actually doing its vacuuming. You can open an app. But just very simple examples of if you have the meaningfulness of getting this information gathered in an application, then you don't have to be physically present or you don't have to be an expert. Of course, you can think if you do this on a much broader scale, you can solve much more difficult problems, and we can talk about some of them if you like.

[00:04:45] JM: Definitely. We'll get into that. But talking more broadly, what are the challenges of gathering this state data when we're talking about a conveyor belt, or a light switch, or a dishwasher, or an electric guitar? What are the challenges of gathering the state?

[00:05:01] DD: I think the first challenge is first of all of course to say do we actually have a means to sense an information on a device? If you talk about a guitar, most likely it was never meant to just figure out in what state it is and then send this information meaningful over the internet. Then you have the connectivity problems. So that's the first two challenges, and you could argue the advent of the internet of things came because now it is so cheap and affordable for almost everybody to actually connect your devices.

I'm always joking, IoT was born when you had come together from the internet revolution, which created, if you like, the cloud and the powerful of our data centers, what we call the mobile revolution, where you had the advent of this very cheap little tiny microcontrollers and connectivity modules that you connect everything. Because now it's affordable that you can connect a light bulb, or you can connect a guitar. Because if you had to add additional electronics which caused a few hundred dollars, why would you connect a light bulb? But if you can do this for a few cents, then it might be meaningful. So the challenges were is there a cheap enough hardware available which can sense the state of any type of thing and can you have connectivity, which allows you to meaningful and cheaply also connect it to the internet?

[00:06:15] JM: My understanding is that the proliferation of smartphones has led to a decreasing cost of these kinds of small components that allow us to connect, for example, a light bulb. Can you give any perspective on how the costs have dropped for integrating these kinds of components into previously non-smart devices?

[00:06:37] DD: That's about areas of magnitude. I started my career in mobile phones, or communications. When I was a student, I mean that was the hot thing. I did my PhD in this area, and the first smartphone – So at that point in time we called them still phones, or mobile phones if you remember, they had the size of suitcases. You had to carry them and they were on the order of magnitude of tens of thousands of dollars. I don't know whether you remember, they were these – You were supposed to be a fancy businessman if you had this really bulky device next to your ears.

Then there came this revolution of everybody wanted to have the mobile communications, and I worked for a company called Nokia, and we sold at the end billions, literally billions of those devices brings the cost down from the entire mobile phone to something like $10 to $15. In order to do so of course you needed all of these electronics, all of these transmission technology, first of all, getting much small and much cheaper. If you're going to a consumer and instead of having a few thousand devices sold but you talk about tens of thousands, millions, hundreds of millions and even billions, then you get the price down. That is then up to two orders of magnitude that the prices were decreased.

As I said, the connectivity module in the beginning, you mostly likely had to buy more upfront of $500. Today, you can get something which is called a system on chip. Think about that there's a tiny little device which looks like a chip but it has integrated memory, sensing, connectivity, some computing power and you get this for a few dollars, and if you really go down to the absolute smallest and cheapest, then we talk about sub-dollar space of tens of cents now. So that was the revolution of mobile, which democratized connectivity for everybody. That's why today, everything can be connected, including a light bulb.

[00:08:20] JM: Most of the listeners are software engineers. They may not be familiar with that term system on a chip, and there's also the term microcontroller, which is another device that you might find in an IoT device. Are these different things? Can you define the term system on a chip and the term microcontroller?

[00:08:37] DD: Yeah. So most software developers might be familiar with something called the CPU, a compute unit, that's the chip, which is in your computer or in your laptop and in your mobile phone. That's a pretty powerful chip which can run an operating system, whether it's Linux or Windows or Android. A lot of people know about Intel, which is producing them at scale. Those are general comp units, and they are pretty expensive.

If you want to do much less compute-intensive things, like just sensing or having a thermostat or controlling an LED light bulb, what you would require is much cheaper cost, and that is called a microcontroller unit. Think about this, that is a much smaller version of a general compute unit, which is purpose built, which has much less power in that sense that is has less megaflops or compute cycles and it also has much less memory. For those compute units which are called microcontroller units to work, you need specialized software, which is either written directly in C or it's called a real-time operating system.

The idea here was if I can have a cheaper hardware with less memory, then of course I need specialized software so that I can make this run. The real issue here is it's of course cost. Why would I if I want to just have a light bulb and LED, why would I need to put a generic, very expensive compute unit into this with an operating system. So rather, give me the minimum capabilities of — in other it's called a digital signal process, a DSP, which then can get it affordable. That's the biggest difference. Think about CPU and a computer. MCU, much cheaper, but therefore also much less powerful with much lesser capabilities, but then you can get a light bulb at price level.

[00:10:17] JM: Amazon builds an operating system for microcontrollers called FreeRTOS. Can you explain what that operating system does?

[00:10:26] DD: So, yeah. Amazon actually took stewardship of real-time operating system called FreeRTOS. That was originally invented by a gentleman called Richard Barry more than 14 years ago, and Richard invented what is called a FreeRTOS by [inaudible 00:10:41] is wild. It's actually very difficult to program these microcontrollers. You need to be an expert as I said before, because you need to understand the intricacies of these smaller microcontrollers because they are not really powerful.

So in order to get most of it, you need to really be a guru in understanding what type of registers do I had to program for? The way to get out of this was to build an operating system for those microcontrollers, and these microcontrollers, they're very often used in real-time environments. Think about an engine for your car where it's very important the fuel gets injected, or any other applications which have what we call real-time requirements.

So the difficulty was to build an operating system, which also has a notion of time and can guarantee certain execution cycles. The world's most used real-time operating system kernel is called FreeRTOS, and we took stewardship of this operating system, and Richard Barry is now an employee of Amazon Web Services, and we extended the real-time kernel of FreeRTOS with capabilities specifically made for IoT, which were you want to connect those devices. So you needed a connectivity stack on top of the basic scheduling functionalities of real-time operating systems, or you wanted to also make it updatable. So how do I update this software, or how do I secure it?

So we enhanced the already existing kernel functions, which is just basic scheduling for software developers with connectivity locally, connectivity to the cloud, updatability and security functionality, and that is now called Amazon FreeRTOS and it's still, as this name indicates, a free operating system, because it's open source under the MIT license.

[00:12:20] JM: We've done a lot of shows recently about Kubernetes, which is a cluster scheduler, a cluster orchestration system for – I guess it's not exactly a schedule, because you build schedulers on top of it, but it's an orchestration system for connecting different nodes together or different containers on those nodes together.

When you're describing FreeRTOS, it sounds like a single node operating system, maybe like a Linux distribution for example. Can you tell about like the state of wiring together these different components? If I have a factory with all of these different components, how are these components communicating information with each other? Is there like a centralized server that they all need to communicate with? Do they all talk to the cloud and then the cloud talks back to them?

[00:13:05] DD: Okay. First, maybe a little correction about FreeRTOS. FreeRTOS is typically used on a single kernel and specifically on a microcontroller kernel, and it is much smaller than the Linux kernel. Even if you think about embedded Linux, embedded Linux still needs megabytes of RAM and a FreeRTOS kernel needs kilobytes. We talk about here three orders or magnitude. I just wanted to clarify this upfront.

[00:13:28] JM: Sure. Yeah, thank you.

[00:13:29] DD: Think about Linux, this is a really big operating system, a real-time operating system, really, really super small. I think to your main question, how does it all work? Yeah, it's a complex environment. There is not one architecture that fits it all. How you can think about this is if I have small little sensors, use the factory example. I have sensors in a machine, from temperature sensors, or speed sensors or any type of sensors, they typically run such real-time operating systems because — or [inaudible 00:13:56] control because they have very dedicated single use functionalities. Then of course if they are connected because you have a connectivity capability, then they are typically connected locally on a local connectivity network, and these can be local Ethernet, or they can be via Bluetooth or anything else.

Then you try to aggregate this in bigger, more powerful devices, like a gateway, and these gateways then they run typically on operating system, whether this is Linux, or Windows, or even Android. On that type of operating system based devices, you can have more logic. For example, we at Amazon, we provide something called AWS IoT FreeRTOS for these microcontrollers, and for the bigger operating system something called AWS IoT Greengrass. That's a runtime which gives no functionality about aggregating all of these data coming from the sensor locally, controlling the machines, doing something which we call pre-processing of data.

Then you can send the most meaningful data into the cloud for even broader processing. For example, if you would like in the factory to understand how do you optimize my logistics, or you would like to understand how the factory output from one factory is managed for the input of another factory downstream. That is the architecture. So think about it, various model devices, microcontroller-based, real-time operating system controlled connecting locally to bigger devices, which can do more logic. Typically those bigger devices then control to the cloud where you can do then the magnitude of compute and control.

[00:15:27] JM: Let's say a customer opens a conversation with you and they say, "Look, I'm operating a candy bar factory and I want to get to a point where I have a really futuristic candy bar factory where everything is connected. I've got predictive maintenance everywhere. I can control everything from a dashboard." But today I just have kind of a state of the art factory, which is some things are smart, some things are disconnected. I don't really have it in a great like centralized organizational state. What is the strategy for the candy bar factory owner for beginning to make their way to the candy bar factory 2.0?

[00:16:07] DD: I think first question a candy bar factory owner would have to answer, "What is that you would like to achieve?" Because you don't just connect everything for the sake of connectivity? What is it what you would like to do? Would you like to save time in producing your candy bars, or would you like to produce them cheaper, or is there something where you want to increase the quality?"

The idea of IoT is not the technology itself. It's always do you want to get a bigger outcome? That was the question you asked me in the beginning when you said, "Hey, what bigger problems can I solve [inaudible 00:16:38] state?"

So whatever we do, whether it's a candy factory or even much more complex environments, is what is the actual business goal? So assume the candy bar factory would now say, "Okay. I would really like to do a fast production line." Then you understand, "Okay. That's the target. Can I analyze how today's production is done and what could I do in order to optimize this?" It's a very different question if you would say, "No. What I want to do is I want to have a different quality control, because then time is less of an essence."

So I think the most important thing to learn about IoT, try to first figure out what is it you would achieve, because depending on what you want to get as an output, you want to have very different approaches, some of them much cheaper and faster, some a little bit more complex, because they are more work-intense. Does it make sense?

[00:17:26] JM: Absolutely. So we've seen these videos, like the candy bar commercial where it's like the chocolate is drizzling over the candy bar, and then the candy bar is like making its way down the conveyor belt. So maybe there's some kind of optimization that we need to do specifically between the chocolate drizzling machine and the conveyor belt machine.

So if we want to instrument those two devices with some sort of connectivity on each of them. We want to be able to sample the data that's coming off of each of them. What do we need to do?

[00:17:58] DD: The first, and it sounds actually almost mind-blowing. The first thing to understand is that typically the different systems in the factory are built from different manufacturers. Believe it or not, state-of-the-art today is that many of those are disparate and actually don't talk to each other. So the first question would be to do, "Okay. If I need to understand the output for," as you said, "the chocolate drizzle on the one side, two for the other machines. Okay, can I actually connect those in a meaningful way that I understand the output from the one as the input to the other?"

If they come from different manufacturers and if they do speak different languages, and a language in computer means they have different data formats, they have different schemas, they might not even have the same input/output. That's the first question you would have to answer, and it's okay. Can I just send the output? Can I just get it out of the I/O channels of my existing machine? Can I put them into a standard data format and can I then make sense of it?

That would be the standard approach, and it's typically – It's a very good question as well, because one of the biggest questions that we always face talking with our customers is just the simple understanding, "Where is my produce in the factory?" Apparently, it's still very handily solved today. It's unbelievable. It was unbelievable to me that when you go in so many factories, that the control of where [inaudible 00:19:19] or where are my parts. It's still very much done by hand based on people's papers and pallets, which you are just moving around. There are very few factories today who have actually systems who can tell you at any given point in time where are those produces and what are the different inputs.

So that seems like a career problem, but believe me it's not, and just solving, understanding where your produce is and what is the backlog from one machine to the other. If you can just optimize this, already has significant improvements in throughput, and therefore the yield of a factory. We talk double digit percentages here when you think about factories really, really, really a lot of money.

[SPONSOR MESSAGE]

[00:20:05] JM: HPE OneView is a foundation for building a software-defined data center. HPE OneView integrates compute, storage and networking resources across your data center and leverages a unified API to enable IT to manage infrastructure as code. Deploy infrastructure faster. Simplify lifecycle maintenance for your servers. Give IT the ability to deliver infrastructure to developers as a service, like the public cloud.

Go to softwareengineeringdaily.com/HPE to learn about how HPE OneView can improve your infrastructure operations. HPE OneView has easy integrations with Terraform, Kubernetes, Docker and more than 30 other infrastructure management tools. HPE OneView was recently named as CRN's Enterprise Software Product of the Year. To learn more about how HPE OneView can help you simplify your hybrid operations, go to softwareengineering daily.com/HPE to learn more and support Software Engineering Daily.

Thanks to HPE for being a sponsor of Software Engineering Daily. We appreciate the support.

[INTERVIEW CONTINUED]

[00:21:29] JM: You mentioned earlier this model where the data coming out of these devices is often aggregated in some sort of hub that's sitting inside the factory and then maybe the hub is communicating with the cloud. Can you describe the software architecture that we might to want to hear in a little more detail?

[00:21:48] DD: Okay. So for most of the factory systems, the inconvenient truth is that a lot of these data never leaves the factory. Today's system use something what you would in layman terms call something like a flight control system. They are called operational historians. Think about this, there's these local databases, actually time series databases, which is small as getting all of the input signals from both devices and track them in an immutable way, and they're again proprietary systems.

Today, many of those, if not most are actually not connected to the cloud. So the first question again is, "How can I access those and get the information out of this system?" There are different type of standards in the factory, one of them called as OPCUA. Think about this as a connectivity standard, locally, plus a data format standard, which more and more manufacturers adhere to.

So there's another idea and saying, "Okay. I have this so-called OPCUA service. Can I use them as aggregation point for the different machines, and then from there go to the cloud." We at Amazon have a new service, which we launched last year at our Reinvent Cloud Conference, which we call IoT Sidewise, and the architecture is as follows. The idea is you use a gateway, which you typically normally have anyway on your local network. You have software on that gateway. We call this the Sidewise Connector Software, which now logs in locally to those databases.

The historians or the OPCUA servers, it can detect from one console in the cloud all of the input signals, and here we talk about hundreds of thousands of data streams which you can have in a factory from all of those sensors and can aggregate them and send them to a data lake into the cloud, which is typically then again in the Sidewise case, a time series database, because it's time series data.

But from a manufacturing perspective, you're not interested in the individual's model sense readings of thousands and thousands of sensors you have on machines. What you are interested in is more operational metrics, for example, an OAE metrics of the operational deficiency.

So what then these systems do like Sideways that you can then define how do I combine these different sensors coming out of these so-called flight recorders or historians then in the factory and then calculate operational efficiency metrics for dedicated machines or ideally for the entire factory, because that's the important metrics that the operational personnel need to understand, whether their machines are working correctly and what they need to do in order to improve. So that's the hierarchy that you can put into place in order to solve those problems.

[00:24:22] JM: Let's give another example. So let's say I'm operating an oil rig in the middle of the ocean. This might be called an edge deployment, and I want to do predictive maintenance. I want to know when different components on my oil rig are going to fail before they fail. This is a very common use case. How does the story change now that we're in an edge environment?

[00:24:44] DD: I'm laughing a little bit. I think we need first to understand what you mean edge, because for us – And again, excuse my bluntness here or my ignorance. When we talk edge as a cloud provider, edge means everything outside of the cloud. A factory is also an edge. Your home is an edge. But I think what you mean in an oil rig is what if the edge or means the deployment is not directly connected in the Internet. Is that the scenario that you're after?

[00:25:10] JM: Yeah, or transient connectivity.

[00:25:12] DD: Okay. So when you say – I don't know. I mean, a mine or in an oil rig. I don't have direct internet connectivity. How would that change then from an architectural perspective and how would you deal with that? Is that the question?

[00:25:22] JM: Yeah, definitely.

[00:25:23] DD: All right. I mean in essence, what that means is you need to have a lot of the data handling locally, because you can't send the data to the cloud to make decision making. That means you have to find a means that you can locally collect, locally aggregate, locally compute and then also make local decision making. That's of course something which has always been done, but it will always – We try to locally optimize.

What have now changed in the internet of things space, and even with those completed local scenarios which you called an oil rig or which can be in a mine, there's now a technology which helps even improving in that situation. Because we, for example, have technology. Think about a really big appliance that you can put down into these scenarios, which is called a snowball edge. Think about a glorified thumb drive, which ruggedized, but you can have tens of terabyte of storage which you could put down, which is then collecting the data and then you physically move those devices from the [inaudible 00:26:23] into the cloud for ingest. That's one way so that you can do post-processing to optimize.

Another way is of course if you have used that data in post-processing and learning then about, “Okay. How does such an oil rig or how does such a mine function,” and you can do optimization whether algorithmic optimization or a more advanced one, which we call machine learning and then find better models for decision making. Then you can deploy those models back to the edge.

I mentioned a software which is called IoT Greengrass, which is running on these gateways which are a little bit more powerful has the ability to take the logic that was calculated in the cloud, in a machine learning model, and then you can deploy this to the edge. The beauty is it takes a lot of compute to train a model, but it takes a much less compute to actually use the model for additional prediction of what we call inference.

So that means if you find a means to collect the data on the oil rig or locally and then get it into the cloud and by doing so optimize a model and bring them back, you can still get better decision making locally. Then of course, even today, even if you’re in an oil rig, they still have a little bit of connectivity. It’s called satellite connectivity.

So what you then try to do is to figure out what are the additional value adding information that you can aggregate pre-process and then send via satellite to the cloud so that you can still do have control over the assets from a central control point, I mean in an oil rig of course. Security and just the safety is one of the most important issues. So you would put all the security and safety relevant systems that you can still see them online even though they are mainly managed locally.

[00:28:06] JM: It’s a beautiful description. So if I understood correctly, let’s say I’ve got this – I like the mining example, or actually I like the oil rig example, because it’s potentially satellite accessible. But like maybe – So first you would take the snowball edge, which is this hardware device that you can just bring to your on-prem deployment. You can suck a bunch of data from your oil rig on to the snowball edge and then you can mail it back to the cloud and then the cloud provider ingests it and then it trains a machine learning model. Then you can deploy that model, which will be much smaller. Maybe you deploy it via some satellite internet connection or some other internet connection, some transient internet connection that you managed to get, and then you can deploy that model.

Then you also described the fact that you do have some connectivity, perhaps through satellite that would allow you to do some control over the oil rig. Could you describe like what are the patterns for controlling these kinds of edge deployments. Do you typically have a software engineer that's onsite at the oil rig that is looking at the results of a machine learning model and they're making control decisions on there or do you have somebody who's remote and they are seeing the data that's coming in through the satellite and they're making the decisions?

[00:29:24] DD: So that really depends on what is the actual deployment. So you can't generalize this. Why I don't like the oil rig scenario is because oil rigs are really complex mechanisms. You have a ton of people sitting there and really monitoring because if something goes wrong in real-time, it's catastrophic. That means you have a ton of specialists sitting there in order to try to prevent anything that goes wrong. But it's still the right example, because you can generalize it in a way and saying is, "Depending on what is your workload."

So if something which is really catastrophic, you will always have people locally there. But there are many others where you explicitly don't want to have personal, because you might have too many of those. What if you try to monitor just in agriculture all of your fields? You can't have a human being standing next to every field and just looking at the sensor readings.

So there you definitely want to have a central control system. Also, when you want to optimize more complex logistics, it is very unreliable just to have human beings all the time looking and staring potentially at gauges and figuring out where you are aggregating this and then making via communications to make sure that all of the information is aggregated.

So whenever you need more real-time information, whenever it's distributed, then of course having those system controllable from one point is really, really beneficial and sometimes it's the only way to do so. So it really depends on the scenarios. But also coming back from your summary, the starting point is always get a lot of data into the cloud specifically when you want to build a machine learning model. But once you have your first instances, then optimizing those models also needs less data and some of them can be done locally and some of them can also be done in the cloud.

Then the beauty is that you don't have to each time bring snowballs back to the cloud. Then with less data, you can improve. So the beauty of machine learning is once you have built the initial model, so training the initial model is very cumbersome. It takes a lot of information, a lot of data, a lot of annotation, but then refining models needs much less data, which of course then the ever improving system that can be done partial locally and then of course partially with much less connectivity to the cloud.

[00:31:36] JM: The listeners are starting to get an understanding for the scope of different problems that are in the purview of AWS here with the IoT stack of technologies that you're building. One technology that I'd like to discuss is AWS Greengrass. Can you explain the collection of problems that Greengrass is solving?

[00:31:56] DD: Yeah. AWS IoT Greengrass is a piece of software which is not running in the cloud, but which is running on premises in our client infrastructure, which can then be a gateway for example. We talked about the CPU-based devices. Think about a gateway, or think about a more complex engine that you have on machine in your factory, which typically can run Linux.

So the question we had to answer for our customers was, "If I have to do local control and I need to do this very fast, because I can't rely on the physics of sending data to the cloud and coming back," which is called – Of course, the speed of light is pretty fast, but sometimes it's just slow for security relevant decision making, or as you said, if I am on an oil rig or in a mine and I don't have the connectivity, how can I bring some of the functionality that we build in the cloud with AWS IoT in the services locally to the edge that I can build those services that they would work also when my factory or my machines are offline?

So that's the starting point and we thought, "Okay, is there a way that we can build something like a software runtime that we could install on existing devices so that you don't even have to buy new ones?" So typically a gateway, which then brings the basic functionality on the local side. Greengrass does exactly that.

Greengrass brings you, fundamentally, four things. It gives you a local messaging bus so that you can have locally smaller and bigger devices connected to each other and then you can control the message flow between them. Without that you have to send any of those messages through the cloud. That's called the Greengrass message bus, which is using an [inaudible 00:33:35] message program, one functionality.

The second functionality you want to do locally is you want to take local actions. So if one sensor sends something, then maybe the control needed needs to say, "Oh! That's a temperature. It's too high. Maybe I need to open a control valve, because otherwise the container might burst." You need to take actions, local actions, and what Greengrass brings you is what we call a runtime for Lambda execution.

In the Amazon world, we call something serverless when you don't have to manage infrastructure anymore. The way of executing code at scale at AWS is called AWS Lambda. It means it's a function, it's a server that [inaudible 00:34:14] service in the cloud normally where you give us code and then we execute the code for you each time there's a given trigger. You don't have to manage underlying infrastructure and you literally only pay for the milliseconds of code execution.

The idea was, "Wow! If this is the new paradigm of how you build applications, can I write the exact same Lambda functions that I can execute them in the cloud and can then execute locally as well?" So that's why Greengrass has this Lambda runtime. So now you can execute code in any given point in time.

The third thing, what we brought with Greengrass to the edge was something which is called data and state synchronization. In IoT, the typical programming model from software is a trigger-based action model, and when you have a pub/sub triggered-based action model, you don't have the notion of state. you have a trigger, you have an action, then you start all over again.

But for many application, you need the notion of state and have a memory. States and data synchronization both locally, and from local to the cloud is the third functionality we brought in to AWS Greengrass. The last one was security, because even if you're not connected to the cloud, you want to secure your local deployments. So making sure that you have a significant authority locally available where your smaller sensors can still be identified and can say, "Oh! That certificate on that sensor is valid," and you can have something what we call TLS endpoints, or transport layer security encryption endpoints. That is all of that functionality on a higher level, that's what Greengrass gives you. So local messaging, local code execution, state and data synchronization and security authority. That's the core of Greengrass.

Then of course once you have those as a deployment mechanism locally, then you can extend it, and we extended it for example so that today you can have now local machine learning inferencing, because now with all those mechanisms, you just make sure that if you have a trained model from a machine learning exercise in the cloud, then by just deploying this model to the edge via Greengrass and having this Lambda capabilities, now you can have all the [inaudible 00:36:19] machine learning deployment to the edge and then local inferencing, or you can have then of course take benefit of what we call local resources that are available only locally, not in the cloud, because if you have a machine, of course the machine might have different capabilities. You might have a camera associated to it, or might have different sensors.

So Greengrass now have something what we call local resource access and then you can access all of the additional hardware elements, which are intrinsic and inherent to your specific implementation. So that is what AWS IoT Greengrass does and it works in consultation with IoT core, the IoT core service and with Amazon FrerRTOS.

So the idea is whenever you have a Greengrass deployment locally, it will automatically synchronize with the cloud with IoT core and it will also automatically would work with everything, which is built from a micro [inaudible 00:37:11] perspective on Amazon FreeRTOS, which connects to it.

So the complicated the infrastructure of I have the small devices, I have bigger devices, which can aggregate. I send all of these in the cloud and then I can manage that. That was the idea why we built AWS IoT Greengrass and that it works together with IoT core and Amazon FreeRTOS.

[SPONSOR MESSAGE]

[00:37:38] JM: Triplebyte fast-tracks your path to a great new career. Take the Triplebyte quiz and interview and then skip straight to final interview opportunities with over 450 top tech companies, such as Dropbox, Asana and Reddit. After you're in the Triplebyte system, you stay there, saving you tons of time and energy.

We ran an experiment earlier this year and Software Engineering Daily listeners who have taken the test are three times more likely to be in their top bracket of quiz scores. So take the quiz yourself anytime even just for fun at triplebyte.com/sedaily. It's free for engineers, and as you make it through the process, Triplebyte will even cover the cost of your flights and hotels for final interviews at the hiring companies. That's pretty sweet.

Triplebyte helps engineers identify high-growth opportunities, get a foot in the door and negotiate multiple offers. I recommend checking out triplebyte.com/sedaily, because going through the hiring process is really painful and really time-consuming. So Triplebyte saves you a lot of time. I'm a big fan of what they're doing over there and they're also doing a lot of research. You can check out the Triplebyte blog. You can check out some of the episodes we've done with Triplebyte founders. It's just a fascinating company and I think they're doing something that’s really useful to engineers. So check out Triplebyte. That's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte. Byte as in 8 bits.

Thanks to Triplebyte, and check it out.

[INTERVIEW CONTINUED]

[00:39:28] JM: In a previous episode we talked about machine learning at the edge and we touched on Greengrass a little bit. That was another show with an AWS engineer. One thing I got the impression of – So one use case that we talked about was like a shipyard, and you’ve got these shipping containers and it’s a really big environment and you’ve got another – This is another scenario where you have disparate connectivity. You don’t necessarily have a great internet connection throughout the entire shipyard, and there’s tons of data that you could potentially be connecting, potentially using.

In this kind of environment where you have like uneven connectivity, you probably have to do things like Bluetooth, or maybe mesh networking. Can you tell me a little bit about the state-of-the-art of networking and what kinds of problems you have to solve?

[00:40:17] DD: Again, networking is a really vast area to discuss and it depends really what you want to do. So you mentioned specifically Bluetooth. Bluetooth is a short-range technology which doesn't need a lot of power. So if you want to build devices that have a batter and that last very long, if you would connect them via Wi-Fi or cellular, then typically that drains batteries pretty fast. So the problem would be how would I charge this?

So you talked about a shipping container. I put a little sensor on a shipping container and I would have to charge this every two weeks, because it runs out of battery. Not a good idea, because I mean putting it on a ship and getting it around the globe takes six weeks.

So the idea is, "Okay. What is the right transmission technology for the right application?" Again, it always starts with the same question, "What problem do you want to solve?" Because if I'm in a factory and I'm stationary, Wi-Fi might be the perfect wireless technology. In most factories, you don't need Wi-Fi because you wire everything because it's even more reliable to have a local internet or Ethernet application. Others we talked already about on oil rigs have to rely on something like satellite.

So the unfortunate thing is there's not a simple answer about that will be the future of connectivity, different type of connectivity, different type of data transmission needs have very different technology requirements. The beauty of course is that we have many of those available. The drawback is that you ideally have to understand all of them depending on how complex your environments are.

But we at AWS, we are not just working with one or two sorts of this connectivity. So there's a differentiation between what we call physical connectivity, which is the actual physical implementation. So would you use a low-energy Bluetooth or would you use a Wi-Fi or would you use an RF? Then you have what is called the logical connectivity, how you send data over those established connectivity methods. By separating them, then you can more or less build your applications independent of the physical connectivity and could always choose the right one that you would like to have for your application. Does that make sense?

[00:42:20] JM: Absolutely. We'll begin to wind down the conversation, because I know you got somewhere to be soon. Just a few more like high-level questions. So I think a lot of the developers who are listening to this show, they think of themselves as like building web applications, or building smartphone applications, and they probably haven't really taken a close look at IoT. I mean, I'm sure there are some people out there. Are there any common misconceptions that developers have about IoT, or maybe you could give some description of the technologies that can get people excited or get them thinking about how to make a transition from being a desktop web developer to an IoT developer.

[00:42:57] DD: First of all, IoT is just a password. I think nobody should be intimidated by IoT. IoT just puts together, if you like, technologies that you now can connect physical objects which originally were never intended to be connected. So when you separate the IT world, which is the information technology, from the OT world, which is operation technology. It means think about machines, light bulbs, anything else. There's this long standing joke that you combine OT and IT, then you get IoT.

But in essence, it's the same type of technologies. I mean, it's connectivity, you need an IT and an OT. It's writing software code, whether you put it in embedded on software or you put it in the cloud. It's different type of code, but it's still code.

What I think is most interesting is to understand what type of problems you could solve, and the reason why I'm so excited about IoT is it can solve some super basic fundamental problems. I'll give you some examples, because that is maybe the most exciting part of why would I look at connecting a physical object? Why would I look at handing all the complexity of these fleets?

We talked about Greengrass, we talked about containers and machine learning at the edge and some of them are very, very complex problems, but some of them are very simple. I mean, we work with a company, it's a startup. Think about a startup called Vantage Power. It's a UK startup, and they are providing the systems before electrifying big buses.

I mean, I spoke with their CEO. He's really excited and was saying, "Okay. My mission is I want to have my contribution to climate change and how can I help? Okay, there's all of these vehicles throughout in the world and they all pollute because they have the combustion engines. What if I can have these very heavy duty vehicles and get electrical power trends?"

That sounds all great and then you figure out, "Oh, if you put them into these big buses and everywhere else, this is actually very immature technology. It works well, but do we know about if something breaks down? Think you have one of these nice little red buses in London and they are running and then something would go wrong and the bus just breaks down? That's a pretty big nuisance because you have to figure out how you get there, the people can then get from A to B."

What they figure out and saying is, "Guess what? Is there a way how could we predict if something goes wrong?" They were doing the following. They were just using all the data that they had available coming from all of the batteries in these buses and they were ingesting something more than a trillion data points and they were building a machine learning model. Here the question was, "Could I find signals in the data that will indicate that one of my batteries will break down?" Because they had measured all of these information and stored them. Whenever something broke down, they could go back and see this, and they were able by just asking that question and using now a lot of data to build models which could predict four weeks in advance if one of these sub-cells in their batteries would break down.

Think about the impact. That means [inaudible 00:45:48] they have installed this now on all of these buses, and that means four weeks before something would go wrong, they get a signal and could say, "Oh! Great, we better schedule something whenever the bus is in the depot." That's a very tangible outcome that you can have, or there's another company, Pentair. Pentair is a filtration company. They build filtration systems for both professional areas like beer brewing, but also for fish farming, and they were telling me about one of their biggest problems that they try to solve, and they were saying, "There's a really big issue for humankind, which is called the protein challenge." There's even a UN program which it's called the 2040 Protein Challenge where they figured out that in 20 years from now, 2040, there will be roughly 2 billion more people on this planet than today. We grow from 7 to 9 billion. But unfortunately, more than 1.5 billion then will not have enough protein to eat on a single day basis, which is humungous problem. They're saying if you would try to feed them with standard meat, the meat production capabilities are incapable of getting there.

So one of the potential solutions for them is fish farming, because apparently fish farming is one of the best contenders at a closed food chain, because there's much more yield, and excuse me if I use the word yield with fish about how do you get protein out of this. Then they are trying now to build with their customers really, really big fish farms. Think about size of airplane hangers, and you can imagine if you have millions of fish in such a big tub or aquarium something goes just slightly wrong and you have an epidemic and there are millions of fish that's catastrophic.

So they are now building systems and instrumenting their pumps and the filtration system and the water just to say, "Okay. Can I sample and get all of the information from chemical content, situations, including temperature –" Imagine everything you can think of and feeding into understanding how do we have to handle those systems so that I can build sustainable fish farming to solving one of the bigger greater humanity problems, which is world hunger? Which were unthinkable, because you could not do this to human beings, because the data that comes is too big. You cannot just [inaudible 00:48:03] understand it.

But going even so far and saying, "You know what? When you look at those fish from the sidelines, they have all of these funny patterns on the side of fish," and I can't see fish apart, but they said if they used cameras and use machine learning, because it's like a thumbnail print, fish can be identifies by their patterns and they have now machine learning models which now you can look at fish and you can identify an individual fish out of millions and can see how this fish is developing overtime and when they are at a certain stage and when they are right for harvesting.

So this are problems that you would have not been able to solve without this technology. Yes, IoT can help you improving your efficiency. You can build better products, which get better overtime, but IoT can also solve problems which were unable to be solved beforehand for the greater good, and if that's something you're excited about, I think that's maybe the best reason just to go and look into IoT and what the technology can do for you.

But at the end of the day, it always starts with the very first question that you started the interview with, which is saying if you knew the state of everything and you could reason on top of that knowledge, which problems would you solve? One said, “I want to solve world hunger.” Somebody else says, “I want to make sure that there’s less pollution.” Another company will just say, “You know what? I want to make more money because I can make better products or I can produce them cheaper.” All of them are very relevant.

[00:49:27] JM: Dirk, thank you for coming on the show. It’s been really fun talking to you.

[00:49:30] DD: Thank you. Thanks for having me.

[END OF INTERVIEW]

[00:49:35] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use, and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plug-ins. Use the value stream map to visualize your end-to-end workflow, and if you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on-the-fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations. You can check it out for yourself at gocd.org/sedaily.

Thank you so much to ThoughtWorks for being a longtime sponsor of Software Engineering Daily. We are proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]