EPISODE 892

[INTRODUCTION]

[0:00:00.3] JM: The service mesh abstraction allows for a consistent model of managing and monitoring the different components of a microservices architecture. In the service mesh pattern, each service is deployed with a sidecar container, which contains a service proxy. These sidecars are collectively referred to as the data plane. Each sidecar provides the service that it is deployed next to with a set of features, such as security policy, rate limiting and monitoring instrumentation.

The sidecars in the data plane communicate with a central module called a control plane. In the control plane, an engineer can operate across these individual services at scale by pushing out updates to them. Kubernetes has made it easier to manage large fleets of microservices and has led to wider adoption of service mesh. Istio is one of the most popular service mesh products.

In today's show, Varun Talwar it returns to the show to describe the state of the Istio project and the process of deploying Istio to a cluster. Varun is the CEO of Tetrate, a company building an enterprise-ready service mesh. Prior to Tetrate, Varun was at Google where he helped found the GRPC and Istio projects.

[SPONSOR MESSAGE]

[0:01:26.7] JM: Continuous integration allows teams to move faster. TeamCity is the continuous integration and delivery server developed by JetBrains. I've always loved the IDEs from JetBrains. I've used WebStorm and RubyMine and IntelliJ. Once you start using any of the JetBrains products, you realize that this is a company that knows how to build products for developers.

TeamCity gives you continuous integration and delivery designed by JetBrains. For most teams, TeamCity is completely free, as long as three build agents is enough for your project. For larger organizations, there is TeamCity enterprise. The listeners of Software Engineering Daily can get

TeamCity enterprise with a 50% discount by going to teamcity.com/sedaily. That's T-E-A-Mcity.com /sedaily.

TeamCity supports most popular programming build tools and test automation systems, version control systems and cloud platforms. Whatever the size of your organization is, check out teamcity.com/sedaily and get started with continuous delivery.

Thank you to JetBrains for being a sponsor of Software Engineering Daily. If you want to try out JetBrains' TeamCity, go to teamcity.com/sedaily.

[INTERVIEW]

[0:02:57.0] JM: Varun Talwar, welcome to Software Engineering Daily.

[0:02:59.1] VT: Thank you, Jeff.

[0:03:00.3] JM: The service mesh abstraction is something that many companies would like to have at this point in 2019, but many companies have difficulty setting it up and deploying it. Why does service mesh have a reputation as being difficult to deploy?

[0:03:19.8] VT: Good question. I think before we answer that, I think there is also a confusion of what service mesh is. The answer to that, I think is also a little bit confusing to people, which is service mesh is not a completely new concept. It's a way to connect all your services securely in a platform layer, as opposed to doing it in silos. It being a platform and people running different kinds of services in different environments, I think it is not something that is very easy to get started on introducing into your organization. That's actually one of the reasons I ended up starting this company, Tetrate, is to actually make it easy to get started, adopt and grow with all the complexity that you have within an organization.

To answer your question precisely in terms of why is it this reputation of it's not easy to get started, I think primarily it's the term is getting popularized via the Istio project that was kicked off at Google. I think it was centered mostly around Kubernetes when it kicked off. That community picked it up well and got started. I think there was interest from many other communities on that concept, but it wasn't really designed for it to be consumable by them. That's one I think reason.

Second is it came – because of it being platform and the concept of it doing a bunch of things that are today done in code, it is rich on features. I think the getting started process of how do you do a specific use case, get it started, was not something that was optimized for, is getting better now with the projects. I think those were two reasons why the perception of hard to adopt grew.

[0:05:33.3] JM: What are the difficulties of setting up a service mesh today?

[0:05:39.3] VT: One thing to understand, this service mesh is a design concept. There are multiple projects, multiple technologies trying to – multiple implementations of that concept, right? The real answer to that question lies in which one we are talking about. Let's say we were talking about Istio, which is one of the more popular ones, then I think the difficulty starts right from what am I doing it for, what problem I'm trying to solve for. I think there has been a lot of this, "Let me just try it, because this is the next new cool thing." Which not starting from a real use case with in hand, is one challenge in terms of how people are approaching it.

The second is that it is in order to – if you're setting it up for greenfield applications, I'm building new things, it is easier to assume this platform and start building. For brownfield, it's not as easy to inject something like service mesh safely, reliably, securely. I think that's the other part of the challenge, making it – the promise is there for brownfield, but because it's somewhat of an intrusive concept, right? You're having proxies take over your data plane, people need it to trust it, it needs to be super reliable, it needs to pass all the checks and balances within an organization for it to be really put to production.

I think the projects are evolving to that. The technology is evolving to that, but there are no clear guidelines in terms of how you make it happen for brownfield applications, what applications, what are the applications you actually do not try this on, or what are they not ready for, versus what they are ready for. I think those are some of the real-world translations, or those are some of those how adoption happens in real-world and what are all the things that need to happen on top of this technology, in terms of best practices, parts of adoption. Those need to happen and they haven't happened yet.

[0:08:02.9] JM: Let's talk more specifically. Let's say I've got a company with a bunch of services. I have my infrastructure on Kubernetes. If I want to deploy sidecar proxies that are envoy to all of those different nodes, what do I need to do to actually get those sidecar set up and deployed?

[0:08:24.2] VT: There's not much in Kubernetes actually. With Istio, you it's you can set up auto-sidecar injection, in the sense that every time a service comes up on a part, sidecar is automatically injected and it takes over – that traffic interception takes over your traffic and then you're up and running. The vanilla case in Kubernetes with your sample apps this is pretty straightforward and easy.

I think the challenge comes when you have specific use cases like, okay, now this service spans not one cluster, but 10 clusters. This service talks to not just within Kubernetes clusters, but outside the Kubernetes clusters, to different other computes. I think those are the scenarios where it's not clearly laid out to people on how to do it.

[0:09:18.6] JM: Does envoy itself have a control plane for interacting with the different envoy sidecars?

[0:09:26.0] VT: Envoy project on its own comes with basic Java and Go control plane that you can extend on and many companies have done that. Yeah, so the short answer to your question is yes.

[0:09:41.6] JM: What does Istio give you in addition to that envoy built-in control plane?

[0:09:47.8] VT: Yeah. Istio was designed for – so one, it was designed primarily for Kubernetes first. It's one thing to just manage the fleet of envoys. It's another thing for your control plane to be aware of the environment you are running in, right? What Istio did was yeah, I'm aware of what services are running in Kubernetes. I'm aware of Kubernetes API server. I'm aware of what those are, what their names are, which namespaces they are living in and it tied all that with the envoys, such that within Kubernetes, you get one holistic experience, right?

Second one was it added a whole lot of features and configuration mechanism, such that you are defining constructs in terms of what you want from your services and not what you want, not configuration knobs of envoy, right? If I want retries and timeout policies for service X to be full in bar, then that's what I define in a way that I'm used to in Kubernetes environment. Istio does a bunch of the translations into the right envoy configurations, making sure it is understood by them, it reaches them, they have those things.

The reality is that this experience that Istio first tried is primarily given for Kubernetes today can be given in other environments as well, right? One of the things that we as Tetrate are trying to do is extend the same concepts to different kinds of compute, right? Because the concepts are very generically applicable for application developers and for where they would want to do is just define what I want from my application and service behavior, in terms of my networking capabilities, runtime security capabilities and default observability that I want for my application. Those are all true, even if you are running some other computes other than Kubernetes.

Mesh is really designed to take a whole bunch of capability that was being written up in code and bring it into this platform piece, make it part of the DevOps platform layer, such that whole lot of cross-cutting concerns around these areas, like reliability and security are built into it. DevOps, network admin, security can all participate together to make this platform running, right?

This is why it's an interesting and a different operating model for companies, because adoption is not just what their technology is, it's how do I adopt it? What do I need to change in my organization to actually make it real? How do I take step one? How do I take go from there to step two? That is the journey that service mesh is going through right now.

[0:13:07.7] JM: One of the goals of Istio is to help you manage security policies. Describe how security policy management works in Istio.

[0:13:20.2] VT: Yeah, people I think loosely say security. Security is a large multi-layered concept. The piece that Istio deals with is the run app security, runtime app security layer, which is all of your securing all the communication between services. There are two parts to it; there is the authentication and then there is authorization.

The first part of encrypting all your requests and traffic from between all of your services without you having to write code in every service to do it is frankly one of the biggest things that still today attracts many people to Istio. How it works is pretty well, straightforward from a high-level, but deep from a technology level. How Istio works is basically as it injects envoy in front between the two services, basically all of your service-to-service communication now is taken over by the envoys on either side. Then you can define okay, I want TLS communication between these services, or in the entire cluster.

There itself, you can see A, how powerful that is, but B, that is also the reason why it can be – it's a little bit intrusive and scope-wise, it can bring a fleeting change, right? Imagine if one of the services there was let's say, running a protocol that envoy did not understand, or let's say it was a storage system where the injection of sidecar caused a performance issue. I think those are the areas where Istio needs to evolve to give more choice and give more incremental adoption, because people try it. They do it. Because you have the choice of cluster-wide, they do it cluster-wide and then something or the other goes wrong and then the user perception is, “It did work,” right? That goes back to your first question of why that perception exists. I think there are powerful knobs exposed to more widely, needs to be more caution behind exposing them.

The other part, to continue this answer, is authorization, which is one of the big needs in organizations is to say centrally, which service can talk to which other service. Within Istio, you can define our back style that authorization policy and you can extend it by other authorization adaptors to define the authorization policies and then and force who can talk to whom, okay. Those are the two big security aspects of Istio.

The concept, if you extend that again to a mesh, which can span across environments, data centers and potentially – and clouds is very, very powerful. It starts to make this application aware networking layer, which is what I like calling meshes. It's this network layer, but now it's aware of all the requests and all that are going through it. I think that is a very, very powerful concept. That is where I think mesh needs to go.

[SPONSOR MESSAGE]

[0:17:14.0] JM: SQL has been around for a very long time. The basics of SQL might not change very much from year-to-year, but the underlying technology that implements those queries is undergoing constant innovation. The Distributed SQL Summit is a full day of talks about building and scaling distributed SQL systems in the cloud.

The Distributed SQL Summit is September 20th, 2019 in San Jose, California. Distributed SQL databases can globally distribute data and elastically scale, while also delivering strong consistency and asset transactions. The Distributed SQL Summit includes speakers from Google Spanner, Amazon Aurora, Facebook, Pivotal and YugaByte DB.

To find out about the latest innovations in large-scale distributed systems infrastructure, mostly with a focus on distributed databases, check out the Distributed SQL Summit, September 20th, 2019 in San Jose.

[INTERVIEW CONTINUED]

[0:18:24.7] JM: You were at Google as a product manager of Istio and GRPC before that. What was Google's go-to market strategy with the release of Istio?

[0:18:37.9] VT: Interesting question. From Google's standpoint, there were a couple of things were coming together; one is Google had the experience of running an architecture like this in their own API platform back-end. A lot of Google API serving works on an architecture like Istio. It's not actually still, but it's a very similar architecture of sidecar proxies.

The second thing was as Kubernetes was getting adopted within community and Google, there were certain gaps which were evident within that Kubernetes community. Joining those two is how Istio came about. My personal experience from GRPC was also, it worked well for people adopting that as for greenfield, like if this is how I build my services based on protobuf and code generation. If I subscribe to that pattern of building services, it worked well. For a lot of existing services where I can't change, or cost of change is too high, there wasn't a clear answer.

That process is then, we came to know of envoy and then pieces came together to form Istio. The rest of the go-to market was mostly to align the ecosystem of partners and support of it. One of the things Istio did was from the get-go, had bunch of industry partners, not just Google. The fact that envoy came from Lyft and then IBM was a participant from day one, and then we had other participants like Pivotal and Red Hat who are the key players in the ecosystem joined behind the project was I think very impactful from how it was perceived in community, how it has quickly risen to the almost one of the de-facto service mesh project of choice.

It’s found its way into most of cloud providers and past platforms already by now. It definitely grew way faster than I thought it would. In some sense, almost too fast. I think that is where now the maturity of project and the popularity are now catching up. They need to go locked instead.

[0:21:12.2] JM: Can you explain that in more detail? What do you mean that the project grew too fast?

[0:21:16.3] VT: I mean, in terms of popularity rate, it became way too popular way too soon, right? More people tried it, but it wasn't ready for such wide adoption and wide set of use cases from start. That's what I mean. It's gotten quite a bit better in the last six months or so, I would say.

Remember, this is in data path technology. People have quite a bit of questions around performance. At the same time, it's also very easy to adopt. One thing to understand is if you look back at Kubernetes when it was launched and what happened to it a year, year and a half from launch, it wasn't very – there was one few key differences. It wasn't this popular this fast. Second, the adoption path was different. You have to actually containerize your app, which requires for you to rewriting and you rewrite code and stuff. This one is more just inject in data plane and get value. It's in that sense, very easy for you to take your step one. That is another reason I think people – there is no entry barrier to it almost, right?

The core project has to be mature enough to absorb all of that interest, from different kinds of people, right? Istio's approach too by developers, by actually operators, by SecOps and InfoSec people, is approached from all these sets of people. They need to have a clear way to take incremental steps to adopt Istio. Incrementally is a keyword, safely is another keyword. I think that's happening more and more. It's going to be in much, much better shape by the end of the year. This is just a sign of how growing project. It's not very surprising. When something is very valuable and easy to get started, obviously more people come through the door, right?

[0:23:25.3] JM: How would that incremental adoption look like? If I'm a company, I want to adopt Istio, or another service match technology, how should I adopt it? What should my proof of concept within my organization be?

[0:23:42.3] VT: Yeah, so it should be driven by your needs. I would pick the one use case, or the one problem that I would want to solve with it and figure out just doing that, just deploying the pieces for it, just picking let's say a few teams and few applications, even maybe starting with one to get that problem solved for one application. I would start that way. Take something, which is as fundamental as for example, canary relays and safe rollouts is another popular feature, I would say within Istio. Where in this world of CICD and faster deployments, you deploy – now you deploy every few minutes as opposed to days and weeks earlier. Now you have multiple versions, newer code getting deployed every few minutes, but you also have to make sure that you are rolling out to your users and you're getting real traffic to it quickly.

If you were to pick, let's say, okay, one application. I want to have – let's say canary release is done, then I would pick for just solving that problem in terms of how the team can make sure that the right configuration, such that they can shift incrementally traffic to newer version, they can see the corresponding metrics and performance for that shifted traffic get confidence in it, dial up the percentage of traffic to the newer version, take it all the way to whatever, a 100% if that's what they want and view back my metrics to say, "Okay, this I feel good about." Complete that journey, just for that component, just for that use case and succeed in it and then move on to the next capability. I think that would be my suggested path to begin.

The other one which we are seeing quite a bit is starting with Ingress. Within an application, there's usually if you're on microservices path, there is many, many microservices that form it.

Then the wider the set of services, more are the constraints around what you want from a protocol perspective, performance perspective, security perspective. Ingress, if you do Istio-Ingress, then I think that people find it as less intrusive, easier way to start. We are seeing that across many companies. Start with that and then loan your configuration bits, get confident in it, start involving your network teams to be comfortable with envoy, have developer and network admin teams collaborate to make sure that this is something they feel comfortable rolling out, they get a sense of how to configure it, they can manage that fleet, they get comfortable with its performance. Then go on to injecting them in sidecar deployment pattern for more service-to-service communication.

[0:27:14.8] JM: That term, Istio-Ingress, can you explore that in more detail? What is Istio-Ingress?

[0:27:20.9] VT: You can deploy, within Istio and within service mesh, there is two concepts, right? There is north-south and east-west. North-south is when you have – you actually, traffic coming from outside. Or another way to people say that is you don't control both sides of the endpoints. It's coming from an external source. East-west is when you control both sides, it's internal.

Typically the first pattern has been served by this API gateway pattern for many, many years. Kubernetes has this concept of Kubernetes Ingress that you can refine a service of type load balancer or group check and expose it outside. Istio basically does allows you to deploy envoy as your Ingress gateway, as well as your sidecar proxy. That's because envoy can act as both. It actually originated as an Ingress proxy at Lyft.

What that means is typically, for north-south you have slightly different requirements than east-west. It's obviously less intrusive. You can just put it in front of a namespace, or a cluster and not in front of every service. Second, there is defined things you want to do in terms of typically you just want to do authentication or some token validation of who is allowed to come in. You want basic metrics of how many requests are coming in. Those are easier, well-known things you can start to get comfortable with envoy on.

Istio as a control plane, you can drive behavior of both the Ingress proxy or the sidecar proxy. Then you can start doing, let's say TLS termination is common. People start to offload termination to envoy, as opposed to let's say the fronting, in many cases F5, or any of those global load balancers in front. That is a common path of how people get started. It definitely gives better behavior than the standard – what standard Ingress that Kubernetes comes with. In that sense, you move a level up from where you were used to within Kubernetes at least. The same things can be done for other computes as well.

[0:29:50.4] JM: You founded the company Tetrate. Explain what your go-to market strategy is for Tetrate.

[0:29:57.0] VT: Yeah, so Tetrate's whole company is focused on service mesh. We want to make envoy and Istio really, really easy to adopt on all infrastructures, all computes, all clouds, on all kinds of workloads. We want to provide – so that is one step that we want to do. In that direction, we have recently created something called GetEnvoy, which you can actually access getenvoy.io. G-E-T-E-N-V-O-Y.io. Which is an easy way for somebody to just get a certified build of envoy and get started.

That is the starting position and we get called on, extend that to have more tooling around it, so you can operate a set of envoys. We want to do something similar for Istio as well, make it really, really easy to incrementally adopt in steps, again on any compute. Then really for us, long-term vision is to form this multi-cloud networking layer, which is application-aware, which has inbuilt security, which can adapt to most all of the workloads and provide a management layer on top, which developers and operators can use. It can fit into the infrastructure of different companies.

In terms of, yeah, go-to market, we are early. We are building this product out with certain design customers, rolling it out. Now we are seeing more and more companies coming to us for the same. It's not publicly launched, but we hope to get that done later this year and go from there. Yeah, so the company's about a year and – well, 15 months old at this point, so early days.

[0:31:57.2] JM: There are multiple different companies that are building their strategy around becoming an Istio vendor. How do you differ yourself from the other vendors?

[0:32:07.0] VT: I think one thing is to make – I mean, one there aren't too many vendors here in this space who really understand on running Istio well. Second, in terms of differentiation, right? At least I think of mesh as a large platform play, where it needs to adapt to traditional workloads, as well as modern workloads. One of the things we are doing is extending Istio to traditional workloads as well, which is VM bare metal as well. Making it really, really simple and easy on those environments as well. Because, I think for people, for organizations to help them in their journey of modernizing, I think it is not only the right thing to do. It's actually very helpful and it aids in their acceleration to modernization by having a layer like this available.

I think that's something different. Envoy, for example is a very small, but pertinent example in that direction, which is how easy it can be for you if you just have a VMware node to just get envoy up and running and front it with your application. Those are baby steps in that direction. Yeah, I think that's one. The approach would be one. I think, second is the team. I mean, just the whole team happens to be a set of people who actively created, or contributed, or are maintainers of these two projects. That's the other part.

[SPONSOR MESSAGE]

[0:34:01.1] JM: This episode of Software Engineering Daily is sponsored by Datadog. Datadog integrates seamlessly with more than 200 technologies, including Kubernetes and Docker, so you can monitor your entire container cluster in one place. Datadog's new live container view provides insights into your container's health, resource consumption and deployment in real-time.

Filter to a specific Docker image, or drill down by Kubernetes service to get fine-grained visibility into your container infrastructure. Start monitoring your container workload today with a 14-day free trial and Datadog will send you a free t-shirt. Go to softwareengineeringdaily.com/datadog to try it out. That's softwareengineeringdaily.com/datadog to try it out and get a free t-shirt.

Thank you, Datadog.

[INTERVIEW CONTINUED]

[0:34:58.8] JM: Do you find yourself competing with other vendors for deals, to help migrate them to – or I guess, I should say onboard them to Istio? Because my sense is that a lot of enterprises do want this service mesh thing, but they need a lot of help to get there. There's a lot of room to differentiate and compete for deals on the access of how deeply you're going to help an enterprise with the onboarding process. Do you find yourself competing for deals with other vendors?

[0:35:33.9] VT: Frankly, right now, no. I mean, as a startup at this stage, I mean, I think the interest we are naturally getting is enough to keep all of us absorbed and busy. There are actually more people who want help than I can help today. I have not seen too much of competition yet, at least that for me personally. Obviously, my data set right now given that I am small is small. Yeah, I'm not seeing that as a challenge yet.

[0:36:07.2] JM: Tell me about your interactions today with prospective customers and people who have become your customer. What are the problems that they're having and what do they want help with?

[0:36:20.2] VT: Yeah. Primarily you're right in the sense of how do you help me onboard to Istio, given my infrastructure, given my constraints, given my plan of modernization, given my journey to cloud, given how I want to increase my developer agility, ops agility? This is the frame with which customers come to us. That's I think common for many of them. Really, it boils down to having any other thing in this world, like prioritizing and having a plan and having a partner who can help you execute that.

Like I said, we are a partner to a few in helping execute that. That journey is very – there's a lot of learnings coming into that that we are trying – we are building into our projects and products. I think that is extremely fulfilling in terms of what – getting those learnings and putting them into the product layer. It's not easy, right? Productizing here is hard, given the heterogeneity that people have and the scope that we are dealing with here. It's not an easy layer to really productize.

I think, service mesh will go – it's definitely early. It is evolving from interest, to adoption, to product, to platform. It will go through that over the next, probably five years or plus. If it goes to the platform stage where for an organization, you can assume this layer exists and developers can just build on top and all of your organizational workflows are built around it, I think it's humongously powerful, in terms of what it does for your agility across the board. Because one thing is with all these microservices and all these framework things is like, yes, there is – developer agility is promised. You forget about the network. You forget about security. Those are real – they need to grow lockstep with your developer agility, with your DevOps model. They need to evolve to a similar model.

Once you do that, I think you can adapt your networking and security to this model of more dynamic workloads and cloud world. I think you can find to save a lot of hours that are today spent in just understanding metrics of a service, or understanding health of a service, its dependencies and all of that. A lot of that stuff can just be done, assumed as part of the platform. That is very, very powerful.

In some sense, I like to think more meta senses, like networking is cool again in terms of – because I think of it as application of where network brings me to the Sun Microsystems line of network is the computer. It is somewhat actually becoming more true again now. To me, a lot of intelligence is now going into the links than the nodes. I don’t know if you can think that way. That is the reason of why –

As you go to more distributed apps world, the more is demanded of links, the more in-depth power is needed in links, the more intelligence is needed in links, so it seems – that part to me is super exciting and is promising for this space.

[0:40:01.0] JM: What's the hardest part about building a company in the “cloud native space”?

[0:40:08.3] VT: It's a good question. Well, certain things in company building are hard period, no matter what company you're building. Those are generally true. The hardest part is generally attracting and retaining good people, so we think. That's my number one on the list from my short experience of 15 months as well. Specifically to cloud native, I think people, there's a few misconceptions in terms of even what cloud native is. Again, I think people sometimes assume cloud is equal to cloud native, which is not necessarily the case. You can have cloud as it exists today, got us to provisioning agility. Yes, one-click, me not maintaining data centers and one-click provisioning of machines and stuff.

Cloud as cloud native, or this notion of I can deploy my application and make changes to them at a rapid pace, deploy them where I need to based on the cost, performance, security, constraints, and almost think of data centers and cloud as the sea of compute. I can place them where it's right to place them and write as a function of as I was saying, your cost, performance, security and so on.

If I can do that, then that to me is – that is more how I think of a cloud native style operating experience, but more centered around applications, not centered around infrastructure. I think I'm a company's point of view, that is what they primarily care about, which is how do I get my applications to be more reliable, measurable, secure with my org structure? Then where I deploy them is somewhat a secondary concern, right? How I do that is something that should be front and center of my experience and I should be able to do that independent of compute, independent of the cloud I choose, or data center I'm running on, right?

My thinking is a little bit different, I think from what how people think of this. To me, I think this is why if you look at the missing links to get to something like that from where we are, I think you'll be able to relate why I think mesh and its future towards this application, or where network layer is a pretty important missing piece in this future. That's what got me even excited to start this, so let's see how far we go.

[0:42:51.6] JM: The first company to come to market with a service mesh product was Buoyant, with the Linkerd service mesh. How does Linkerd compare to Istio?

[0:43:02.9] VT: Linkerd is a great project. In fact, both William and Oliver who I know was doing a phenomenal job. In fact, service mesh I think is a name – in fact, the credit even goes to them, because I think they were actually one of the first ones to come up with.

I mean, in terms of specifics of Linkerd, yeah, I mean, I think their latest work over the last six months or so is pretty phenomenal, in terms of the experience and speed that they are bringing, which is great. I think more options is better for consumer. One thing which is personally to me, I think, envoy data plane and envoy's adoption is growing pretty rapidly. The community which is adding capabilities and maturity to envoy industry is growing pretty rapidly, an ecosystem there is forming at a faster rate. I think that, I would see, if I were Linkerd I would see as a challenge. Yeah. I mean, otherwise, both William and Oliver are doing a pretty fantastic job.

[0:44:06.8] JM: What about from an engineering perspective, what are the design decisions that Linkerd does differently than Istio?

[0:44:12.7] VT: From an engineering point of view, look, I think there is – going back to the core data plane, I think, so they've launched a new data plane which is Rust-based, which is fast, which is what is bringing a whole bunch of data plane resource utilization games that they've been talking about and speed, which is great, right? I think. I think for this to be widely adopted beyond ecosystem purely from engineering, I think you need it to adapt to – there's a whole bunch of things you need in the data plane, because this whole concept is about centrally managed and locally enforced, or centrally managed and locally executed, where you can define centrally what behavior I want, but locally running proxies can actually enforce those.

For that locally running proxies to adapt to different workloads, you need protocol additions and envoy is already adapting to a bunch of non-HTTP protocols and more are getting added. Second, performance [inaudible 0:45:15.4], just lot more people working on tuning of performance of envoy. Third, in a world of this, you want to have specific injected behavior, like I want to check on whether a given request payload has – or forget payload. Or, even header has any SQL string, or I want to do something on the payload and I want to write certain custom behavior, which is pretty common to, I want to do certain checks on whether this request is allowed to do. I read something, inherit that in payload and make that decision.

These are just examples in authorization space, but this need of customizing the behavior of these locally running proxies is another from an engineering point of view, a need. Then that is also a place where I think envoy with its efforts is a little bit ahead and growing faster. For me personally and obviously for our company, we are placing our debt on envoy and Istio. Frankly to me, I think the larger vision of enabling this service mesh for the organizational impact is way more important. Let's say, five years from now there is an even better, faster magical data plane that we do, or someone else does, great. We should adopt that.

[0:46:42.8] JM: Do you think that the service mesh war is winner-take-all?

[0:46:48.9] VT: No, I don't think so. For a while, it will be – there will be many implementations. In the long-term, probably one is going to win, I think. The reason is I think, they'll be – again, because the core plumbing is off the mesh in terms of the data plane and the protocols and its performance characteristics, it having all the buy-in from all the organizations, it being compliant, the set of things that need to happen to the core plumbing for it to be widely adopted is a pretty high barrier. I think it'll be difficult for multiple things to cross those barriers.

Where you'll have choice, so in that sense, one will win. Where you'll have choice is the platform piece, where you can have adaptors and different companies. Let's say for example, I want to have a very – I have a vendor when I used to do threat detection, the specific – this whole bunch of companies that help do that. I want to take signals that are coming from this layer, but I can then plumb in my third detection software, awesome. There will be companies providing that, and so on like this. I think there, you'll have many, many choices. Same for let's say, even observability or security. On the core foundation, I do think long-term, one will win.

[0:48:14.0] JM: Varun, thank you for coming on Software Engineering Daily. It's been really fun talking to you.

[0:48:16.8] VT: Thank you, Jeff.

[END OF INTERVIEW]

[0:48:22.3] JM: At the beginning of 2019, we had problems with Software Daily, which is our custom-built website and mobile app set. The website was not engineered properly and our iOS app was buggy. Everything needed redesign.

To help us refactor our cross-platform application, we brought in Altology. Altology is a full-stack software engineering firm that helps innovators build worthwhile products. Altology will help you get your project or your company to where you want them to be. They can rescue your project, they can augment your team, they can help you get a new version of your product out the door. If you're building a brand-new product from scratch, Altology can also design and develop web and mobile products that are brand new.

The Altology team is entrepreneurial, they’re design-focused and they're able to work across the stack. To get help with your engineering projects, check out Altology today by going to altology.com. That’s altology.com. Thank you to the Altology team for helping us get Software Daily to where it is today, and for being continued friends of the show. If you need help with your application, check out altology.com.

[END]