FAQ

- ***How to describe Ambient Mesh in a sentence?***

Ambient Mesh is a new data plane mode for Istio that's designed to feel ambient. And by that, its mean, from the perspective of an application running in Kubernetes, Istio is just part of the network. It doesn't have sidecars or it doesn't have to kind of modify its configuration to use it. And I think that gives it simplified operations and a bunch of benefits.

- ***At 10,000 feet, how is Ambient different to istio sidecar based model? How does it work as a model?***

Ambient Mesh as three core architectural changes in Istio. **First**, it's sidecar-less. In Istio, Our application sends traffic, IP Tables captures it, sends it to a sidecar, and the sidecar forwards it. In Ambient Mesh, your application sends traffic, it leaves the pod, and it's captured by a node local proxy called the ztunnel, which forwards it. So there's no sidecar. It's more of a node proxy.

Second — the architecture is layered. So in Istio, everything Istio does is implemented in the sidecar, from encryption all the way through L7 functionality. In Ambient Mesh, we actually break the architecture into a secure overlay that's just responsible for basic encryption, identity, and routing of traffic. Then on a per-namespace basis, we can deploy what's called a waypoint proxy, which gives us full L7 functionality, so everything that you're used to with Istio today.

third piece is It interconnect all these pieces using HBONE, which is a new tunneling protocol that — well, it's not that new — but it's a new tunneling protocol that is developed for Ambient Mesh.

- ***What do people actually use Service Mesh for? And how did that factor into how Ambient was designed?***

When you actually look at usage statistics, at least within ANZ Bank, a large percentage of our users are just using **Service Mesh for encryption**. And if we look at why people get started with Service Mesh, I suspect that's almost everyone. They've got unencrypted traffic, and they have some regulatory requirement that says you must encrypt your traffic, and Istio is the easiest way to do that. And then, once they have it installed, they start using more complicated features over time. But **basic encryption and identity are kind of the main use case.**

The primary use case of Ambient, being the mTLS encryption, the nice thing about the way that the Ambient architecture works is that if you only want that encryption, we only need to deploy the ztunnels and send the traffic between them. So there's no need to have them go through a waypoint proxy or do any L7 processing at all, which will improve the performance, because it's much more efficient to just tunnel the traffic than it is to terminate it and do full L7 processing.

- ***What problems people have with sidecars as a deployment pattern? Is it just the fact that Kubernetes doesn't support them as a first class object?***

There was a lot of interest in the ability to enable mTLS, for example, in workloads for GKE. And in order to do that, injecting the sidecars is pretty disturbing. The workload has to get restarted. You insert this proxy that is terminating L7. And that L7 proxy can actually be pretty disruptive to some networking stacks. So if the application has a broken HTTP stack, for example, the proxy may actually not be able to process that properly, and it can lead to breakage.

Another issue with sidecars, they do break some workloads. They're also a little bit inefficient from a resource-usage perspective. So if we think, on a particular pod, we have to allocate the RAM and CPU for that sidecar for kind of the worst-case usage that we expect for that pod. In practice, what we found with many customers is they end up having massive resource allocations to these under-utilised side cars. They're very inefficient from a RAM/CPU perspective. And that's something that people complain about.

I found that, once people have deployed sidecars, they work pretty well for those people. But there's a certain amount of work that needs to be done to make sure that it's going to work in that environment, which is pretty different from the ability to just be able to turn them on or turn them off. And so we think that using the ambient approach, getting rid of the sidecars, is going to lower the bar for people to make use of Istio without having to take on all of those architectural changes from sidecars.

- ***With Ambient Mesh we talked about the secure overlay agent. You've referred to it as the ztunnel. What does the Z stand for?***

Zero trust

- ***Istio has long told people it's not secure to share an Envoy proxy between workloads. So have the Envoy maintainers. Why is it safe to share a ztunnel ?***

My position is that it's not safe to share layer 7 processing of traffic between workloads in Envoy. But all the ztunnel does is it receives a byte stream, encrypts it, and forwards it. So a lot of the risk associated with the Envoy HTTP stack doesn't really come into play for us. So I think it's safe. Importantly, in current state its a experimental branch, where the community using Envoy as the ztunnel implementation, but we almost can think of that as more of a reference implementation. And over time, we expect there to be multiple competing implementations, some of which may be Envoy, some of which may not.

- ***<u>Where's the eBPF? when we talk about Ambient Mesh.</u>***

Ambient Mesh is agnostic to how the ztunnel captures traffic. And that will depend on what CNI you're using in the Kubernetes cluster that you install Ambient Mesh in. In the experimental code we released, it's not based on eBPF. It's based on IP Tables.In brief, within Google (primary contributor to this project, uses multiple CNIs that someone can use to run Kubernetes, and on those CNIs that use eBPF, they use eBPF to capture traffic in Ambient Mesh. So think eBPF is cool, but it's just a tool for getting traffic to the ztunnel for us.

- ***<u>In above question we mentioned that networking stack redirects all traffic for participating workloads through the ztunnel. How does it know to do that, and then how does the ztunnel know where to send the traffic?</u>***

The way it knows to do that is, Ambient mesh set up the CNI to know which pods are running in kind of Ambient mode, and it knows which virtual ethernet devices are associated with pods that are running in Ambient mode. And essentially, the CNI is instructed to take all traffic that egresses that virtual ethernet device and send it to the ztunnel pod. So the host CNI is responsible for getting traffic to the ztunnel, however it does that. And then we have a controller — the Istio, basically — programs Envoy, which is how the ztunnel is implemented, to know, hey, if I receive traffic with this source IP and this dest IP, encrypt it with this key and forward it to that next hop.

- ***<u>As we know different CNIs that can be installed on Kubernetes. Istio itself has a CNI. The Istio CNI plugin is installed on each node and the ztunnel is installed on each node. Why are these two different things?</u>***

They seem vaguely related, but they're not really. The Istio CNI — first of all, it's not a full network CNI. It's not intended to be run on a Kubernetes cluster without another CNI that handles IPAM and basic L3 routing and all of that sort of stuff. It's got a fairly limited purpose, which is essentially setting up sidecars and pods. Setting up the routing of traffic within the pod to work with the sidecar.

So you still need the Istio CNI if you're going to run sidecars in your cluster. And it's the best way to do that. The ztunnel has really nothing to do with sidecars, so it doesn't really depend on the Istio CNI. And there's a little bit of a misnomer. I use the term CNI kind of to mean a full, complete networking CNI that handles everything, kind of like Calico, or Cilium, or there's a number of these floating around. But the Istio CNI is a much more limited thing.

- One of the things most of us want out of a Service Mesh is security, is to know that traffic is encrypted between all of the nodes and the workloads that run on them. A lot of us will think about security in the context of VPNs, which is a thing that they are used to in their head, where everything goes through some kind of tunnel. When we talking about mTLS, we understand that the m means that both ends of the connection are verified. But I was surprised when I first learned that mTLS is actually a feature of HTTP, and what we doing is we are taking our existing HTTP connection and we are upgrading the security on it. we are not actually tunneling it. That's the way Istio has worked until today. Now we are taking a tunneling approach. Talking you through the decision and whether or not it was possible to achieve what we wanted to achieve with the existing mTLS approach.

mTLS is pretty narrowly about encrypting byte streams in a specific way. And the property of it that's particularly important to us at Istio is that both ends of the connection, down to the pod, have a cryptographic identity, and both ends of the connection are verifying the other end of that identity. That's what makes it different than regular TLS, where your browser connects to Google. Google doesn't really care who you are. It'll run searches for us. So that's one problem. And Istio community think that is the best way to do encryption, for a bunch of reasons. **There's a separate problem of, how do we move traffic through the network?** And tunneling has a lot of advantages over what Istio was doing before that basically come down to, if you establish a tunnel, you can forward any traffic anywhere without having to care about or know what that traffic is. So now we can forward TCP, or HTTP, or anything. And we don't have to modify it or understand much about it. So that's why Istio community kind of doing mTLS with HBONE while Istio is doing mTLS upgrade without tunneling, which is less good.

- ***<u>HBONE, this is something thats introduced alongside Ambient Mesh, what is an HBONE?</u>***

HBONE is this new tunneling mechanism that we're using in ambient, although it actually has benefits for sidecar as well. And so it's available for that. HBONE is a new standard that we're defining to allow using HTTP connect to tunnel the traffic. So, what Istio community done is they used an HTTP-based tunnel. And so actually, **HBONE stands for HTTP 2-Based Overlay Network Environment.** And so we do an mTLS connection between the two entities that are talking, and then **we create individual HTTP connect streams within that tunnel for any two entities that are talking to each other that are on those nodes**. it allows us to forward that traffic uninterpreted so that it will not break. If we have something like a nonconforming HTTP stack, because we're not terminating that connection, we're just forwarding it through, we're able to just send that through an HTTP connect tunnel.

- ***Is there any particular advantage to the tunnels being based on HTTP at this point, if they're just tunnels?***

To answer this, we need to debate or discuss on do we **tunnel packets or byte streams?** Traditionally, networking, it's very common to tunnel packets. There's GRE. There's GENEVE. There's vxlan. There's a bunch of packet tunneling protocols. It is less common to do what we're doing in HBONE, where when ztunnel receives traffic, it doesn't actually care about how it was literally packetized.

Ztunnel - It takes it, it terminates TCP, it creates a byte stream, and then it shoves that byte stream into an HTTP. So if you're going to send an encrypted byte stream on the internet, there's no alternative HTTP. Like, someone could invent a new protocol, but why? There could be a debate of whether we should use HBONE or one of these network-encrypted packet tunneling protocol. For various reasons, there isn't one that istio community thought met there requirements today, so they went kind of this route of using ztunnel based on "tunnel byte streams not packets".

###############################

Layer 7 based Questions:

- ***The waypoint proxies are enabled between a client and a service, or a consumer and a producer, if we prefer. They appear to sit in the middle of the network as opposed to the sidecar model, where you had a proxy at both ends. What is a waypoint associated with?***

The first thing that happens when you enable Ambient Mesh is that we do this secure overlay. Then we allow — on a namespace basis — to define L7 policies. And so typically, if you were to have an L7 authorization policy, then, for that producer side, you would enable a waypoint for that. Then all of the clients then would, instead of connecting directly to the node that is hosting those producers, then their traffic will now be routed to the waypoint. And then that waypoint will then forward the traffic to the ultimate destination. And the L7 policies occur there.

One of the less obvious implications of this is that it modifies the way that load balancing is done. So currently, in Istio, with the sidecar model, each client does the load balancing to decide which server to send the traffic to. And in the model that we're using with waypoints, all of the clients, if there's L7 load balancing, would be sent to that server's waypoint proxy, and then the waypoint proxy would then distribute the traffic. And in a lot of ways, this is more what we would expect a network to do, because that proxy that's sitting in front of the server has a lot more context about what traffic has previously been seen. So the load balancing should be much fairer in this new model with the waypoint proxies.

- ***One of the things that was brought up as an advantage of sidecars when they first launched was the idea that this was distributed, and we now didn't have single points of failure with mid-tier load balancers. How do we get around that problem? How do we deal with what happens when our waypoint fails?***

Most of the architecture drawings we've been doing have shown a single waypoint proxy, but that's actually not how it will necessarily be implemented. Istio can control how many waypoint proxies there are and load balance among them as well so that we can create some redundancy there to make sure that a waypoint proxy is always up and available. Since the waypoint proxies are just regular pods that are scheduled in Kubernetes, then we can make use of Kubernetes' ability to do auto-scaling.

Waypoints proxy is that they're the layer 7 proxy for a particular namespace. And the simplest way to think about it is, if you deploy a waypoint on a namespace, then all traffic that enters that namespace will go through the waypoint. The logical conclusion of that is that it makes sense to implement load balancing and routing policy for a namespace on that namespace's waypoint rather than at the sidecar of everyone who's talking to that namespace. It's subtle, but what this gives you is actually a much better administrative boundary, where, if a particular namespace wants to implement a particular set of policies, it can deploy a waypoint and all traffic going to it will get those policies, versus in the sidecar model, if I want a particular load balancing policy, everyone who talks to me has to implement it, no matter the cost. And I have to trust them to implement it. So it's subtly shifting how network policies are implemented to be closer toward the server rather than the client, which is how people tend to think about these things anyway.

- ***One of the great things about Ambient Mesh is it's not all or nothing. You can enable it on certain workloads, and indeed, you can interrupt right between Ambient mode workloads and sidecar mode workloads. How is that possible?***

This is kind of part of the magic of HBONE. So we're requiring everyone to implement this much more flexible transport protocol. By the way, aside from Ambient, we do expect that all sidecars will migrate to HBONE over the next couple of releases anyway. But then, once you have that, the Istio controller that has a bunch of waypoints and a bunch of sidecars, and if sidecar sends traffic to a pod that is in an Ambient configuration, the control plane knows, instead of sending the traffic directly to that pod, it sends it to its waypoint, for example. So this is just something that the control plane can do because it knows, for each pod on the network, is it sidecar, secure mesh, waypoint, or not meshed at all? And it routes accordingly.

- ***We always think about this being applied to namespaces. There is a certain implication that means that the way you deploy workloads is effectively each set of applications in its own namespace rather than, perhaps, a namespace for a group in a business who have all of their applications running in it. Does Ambient require you to run things in a way that's compatible with it, or can you apply this model to however people choose to align their objects with Kubernetes?***

The namespace is the configuration boundary. So this isn't exactly true, but in almost all cases, all workloads within a namespace need to have the same ambient configuration. They either need to be all sidecar or all secure overlay or all waypointed. From a security perspective, waypoints actually aren't deployed on a per-namespace basis. They're deployed on a per-service account, which is the Kubernetes concept of a network cryptographic identity basis. So the practical result of that is you don't have waypoints that are sharing keys. Each waypoint is its own identity. Now, they can horizontally scale. You can have five of them with a particular identity, but you don't have one with two identities. So from a security perspective, however you organize it from a namespace perspective, it doesn't really matter. But many other things in Kubernetes, namespaces have an impact on how fine-grained your configuration can be.

- ***What about if I want to control traffic leaving my applications rather than coming into it? Is there an idea of processing egress traffic with waypoints?***

Yes. So the logic about which waypoints to go through and have to go through a waypoint is quite subtle(Intelligent). The algorithm is something like this. If you're sending traffic, if there is a namespace in front of the server, send it to the server. If the client has a waypoint, send the traffic through the client's waypoint. If there's either no server waypoint at all or there's an egress policy. So if there's some policy that says, add this header to all traffic that leaves this namespace, we have to send it to the client namespace. So that's a little detailed, but at a high level, I'll just say, Istio intelligently figures out whether it has to send traffic through one or two waypoints depending on the policy we apply. And it heavily prefers to send traffic through exactly one waypoint if it can.

- ***One of the pros of the sidecar model is that all of the traffic to a pod is intercepted, and thus, I know everything going to a workload has gone through a sidecar. With a waypoint proxy, that may not necessarily be true. How do you stop people bypassing the security functions of the mesh and talking directly to your workload?***

So on ingress, yeah, you do have that guarantee, not on egress. It's actually fairly simple to enforce this with waypoints. So basically, the ztunnel knows that a particular pod that it's operating for has a waypoint sitting in front of it. And if it receives traffic from any pod other than the waypoint, it will hairpin it through the waypoint. So it will send it to the waypoint for processing and then back. So you still have that guarantee, but you're relying on the control plane to enforce that for you.

##############################

Some tough questions about Ambient Mesh

- ***Is Ambient Mesh model is secure compare to existing sidecar based mode ?***

So like with most things in security, there's a trade-off. And so we think that it is no worse than the sidecar model. It does have a couple of benefits. One benefit is that the application and the sidecar are not co-located. So currently, if you have a vulnerability in either the proxy or the application, both can be attacked. So by separating them, we make it so that a vulnerability in one of them does not affect the other.

And then also, for the ztunnels, currently, they are implemented as envoys. But we run them in just L4 mode, so there is no L7 processing that's happening for policies. And so I believe, also, that the vulnerability surface area is going to be much smaller for the ztunnels than the existing sidecars.

- ***Is Ambient Mesh model is faster compare to existing sidecar based mode ?***

The answer is sort of. The experimental code released is highly not optimized. And in early tests with that, foundation team found that actually, the performance, the latency and the resource use — the latency is close to sidecar. It's acceptable. And actually, they have some early promising results that it will result in a pretty significant reduction in RAM and CPU. That's the sort-of part. The code that's out there is about the same as sidecar.

I expect the architecture, particularly for people who just want encryption, to be quite a bit faster than sidecar in the long term. And the reason for this is that they found that the majority of the cost of Istio is in the L7 processing. It's in the HTTP processing. And in Ambient, you go from two HTTP processing steps per request to either one or zero. Now, you pay a network hop to get that one request. But at least Google's network is super fast and it's kind of rounding error doesn't really matter.

So I think the architecture, fundamentally, will be significantly faster than sidecar for the secure overlay and about the same for traffic going through waypoints. And the experimental code, I would say, is acceptable but really not optimized.

- ***That also kind of ticks off, "is it leaner". Is it cheaper?***

The short answer is yes. We expect it to be significantly cheaper for almost all users. And if you think about it, in sidecar basis too, you've got all of these proxies running around eating up RAM and CPU reservation that you're paying your cloud provider for that are mostly not doing anything. In Ambient Mesh, the waypoints are scaled to the actual load that's going through them. And the ztunnels aren't that expensive, because they're just doing it for processing.

The other thing to consider, and we're not quite sure how this will work out, but we just have less proxies floating around that need updates from XDS, and the telemetry captured from them, and the operational weight of the system reduces a little bit for that reason. But we're early in the system. So we're going to learn more about that as we kind of get it closer to production.

- ***Is it easier to upgrade?***

Yes is the short answer. So the upgrade of a control plane will be the same. With Istio today, if you want to upgrade your sidecar, you have to restart the application. That doesn't sound like a big deal, but if you work at a big enterprise environment with a bunch of application teams, some of whom are running stateful workloads that can't be restarted easily, and you have to go to all of them and convince them that they should restart their application, that's a big, painful lift.

In Ambient Mesh, the Envoy proxies can be upgraded without disturbing the application. It's a purely infrastructural change. And the waypoints, in particular, we plan to upgrade them by just running the new version and slowly shifting traffic over. So you don't even have to disturb any kind of currently running traffic at all. So ultimately, it's going to be a much better operational experience from upgrade and installing a new workload, and that sort of thing.

- ***little bit about the things that are lost for people who choose to move to Ambient and why people might still want to stay with sidecars?***

WebAssembly and extensibility is currently missing but will be implement prior to taking it to production. Sidecars fate-share with the application, so you can pay a little bit less care toward kind of maintaining the waypoints and the ztunnels and that sort of thing. You don't have risk of those failing independently of the application. So it's just a slightly different operational model.

The security model is different. Suppose, if we vetted the sidecar model from a security perspective, and it's really important for you that the key is in the pod of the application.

The L7 telemetry will be quite different. So currently, since we have — in side cars, since we're doing full L7, one, we can look at the traffic that's being sent, and also, we can insert information so we could do things like track how long a request takes — and so since we're generally going to be removing one of those hops, we lose that ability to do that sort of tracing.

If you want L7 visibility, you can deploy a waypoint proxy to get that, but we don't really have a good model if you want to have that end-to-end performance, or being able to get the amount of time it takes for a request from end to end. But we can, even with just the waypoint proxy, determine how long it takes the back end to handle a request, for example.