

The New Internet

Avery Pennarun - juillet 26, 2024

<https://tailscale.com/blog/new-internet>

We don't talk a lot in public about the big vision for Tailscale, why we're really here. Usually I prefer to focus on what exists right now, and what we're going to do in the next few months. The future can be distracting.

But increasingly, I've found companies are starting to buy Tailscale not just for what it does now, but for the big things they expect it'll do in the future. They're right! Let's look at the biggest of big pictures for a change.

But first, let's go back to where we started.

David Crawshaw's first post that laid out what we were doing, long long ago in the late twenty-teens, was called Remembering the LAN, about his experience doing networking back in the 1990s.

I have bad news: if you remember doing LANs back in the 1990s, you are probably old. Quite a few of us here at Tailscale remember doing LANs in the 1990s. That's an age gap compared to a lot of other startups. That age gap makes Tailscale unusual.

Anything unusual about a startup can be an advantage or a disadvantage, depending what you do with it.

Here's another word for "old" but with different connotations.

I'm a person that likes looking on the bright side. There are disadvantages to being old, like I maybe can't do a 40-hour coding binge like I used to when I wrote my first VPN, called Tunnel Vision, in 1997. But there are advantages, like maybe we have enough experience to do things right the first time, in fewer hours. Sometimes. If we're lucky.

And maybe, you know, if you're old enough, you've seen the tech cycle go round a few times and you're starting to see a few patterns.

That was us, me and the Davids, when we started Tailscale. What we saw was, a lot of things have gotten better since the 1990s. Computers are literally millions of times faster. 100x as many people can be programmers now because they aren't stuck with just C++ and assembly language, and many, many, many more people now have some kind of computer. Plus app stores, payment systems, graphics. All good stuff.

But, also things have gotten worse. A lot of day-to-day things that used to be easy for developers, are now hard. That was unexpected. I didn't expect that. I expected I'd be out of a job by now because programming would be so easy.

Instead, the tech industry has evolved into an absolute mess. And it's getting worse instead of better! Our tower of complexity is now so tall that we seriously consider slathering LLMs on top to write the incomprehensible code in the incomprehensible frameworks so we don't have to.

And you know, we old people are the ones who have the context to see that.

It's all fixable. It doesn't have to be this way.

Before I can tell you a vision for the future I have to tell you what I think went wrong.

Programmers today are impatient for success. They start planning for a billion users before they write their first line of code. In fact, nowadays, we train them to do this without even knowing they're doing it. Everything they've ever been taught revolves around scaling.

We've been falling into this trap all the way back to when computer scientists started teaching big-O notation. In big-O notation, if you use it wrong, a hash table is supposedly faster than an array,

for virtually anything you want to do. But in reality, that's not always true. When you have a billion entries, maybe a hash table is faster. But when you have 10 entries, it almost never is.

People have a hard time with this idea. They keep picking the algorithms and architectures that can scale up, even when if you don't scale up, a different thing would be thousands of times faster, and also easier to build and run.

Even I can barely believe I just said thousands of times easier and I wasn't exaggerating.

I read a post recently where someone bragged about using kubernetes to scale all the way up to 500,000 page views per month. But that's 0.2 requests per second. I could serve that from my phone, on battery power, and it would spend most of its time asleep.

In modern computing, we tolerate long builds, and then docker builds, and uploading to container stores, and multi-minute deploy times before the program runs, and even longer times before the log output gets uploaded to somewhere you can see it, all because we've been tricked into this idea that everything has to scale. People get excited about deploying to the latest upstart container hosting service because it only takes tens of seconds to roll out, instead of minutes. But on my slow computer in the 1990s, I could run a perl or python program that started in milliseconds and served way more than 0.2 requests per second, and printed logs to stderr right away so I could edit-run-debug over and over again, multiple times per minute.

How did we get here?

We got here because sometimes, someone really does need to write a program that has to scale to thousands or millions of backends, so it needs all that... stuff. And wishful thinking makes people imagine even the lowliest dashboard could be that popular one day.

The truth is, most things don't scale, and never need to. We made Tailscale for those things, so you can spend your time scaling the things that really need it. The long tail of jobs that are 90% of what every developer spends their time on. Even developers at

companies that make stuff that scales to billions of users, spend most of their time on stuff that doesn't, like dashboards and meme generators.

As an industry, we've spent all our time making the hard things possible, and none of our time making the easy things easy.

Programmers are all stuck in the mud. Just listen to any professional developer, and ask what percentage of their time is spent actually solving the problem they set out to work on, and how much is spent on junky overhead.

It's true here too. Our developer experience at Tailscale is better than average. But even we have largely the same experience. Modern software development is mostly junky overhead.

In fact, we didn't found Tailscale to be a networking company. Networking didn't come into it much at all at first.

What really happened was, me and the Davids got together and we said, look. The problem is developers keep scaling things they don't need to scale, and their lives suck as a result. (For most programmers you can imagine the "wiping your tears with a handful of dollar bills" meme here.) We need to fix that. But how?

We looked at a lot of options, and talked to a lot of people, and there was an underlying cause for all the problems. The Internet. Things used to be simple. Remember the LAN? But then we connected our LANs to the Internet, and there's been more and more firewalls and attackers everywhere, and things have slowly been degrading ever since.

When we explore the world of over-complexity, most of it has what we might call, no essential complexity. That is, the problems can be solved without complexity, but for some reason the solutions we use are complicated anyway. For example, logging systems. They just stream text from one place to another, but somehow it takes 5 minutes to show up. Or orchestration systems: they're programs whose only job is to run other programs, which Unix kernels have done just fine, within milliseconds, for decades. People layer on piles of goop. But the goop can be removed.

Except networking.

You can't build modern software without networking. But the Internet makes everything hard. Is it because networking has essential complexity?

Well, maybe. But maybe it's only complex when you built it on top of the wrong assumptions, that result in the wrong problems, that you then have to paper over. That's the Old Internet.

Instead of adding more layers at the very top of the OSI stack to try to hide the problems, Tailscale is building a new OSI layer 3 — a New Internet — on top of new assumptions that avoid the problems in the first place.

If we fix the Internet, a whole chain of dominoes can come falling down, and we reach the next stage of technology evolution.

If you want to know the bottleneck in any particular economic system, look for who gets to charge rent. In the tech world, that's AWS. Sure, Apple's there selling popular laptops, but you could buy a different laptop or a different phone. And Microsoft was the gatekeeper for everything, once, but you don't have Windows lock-in anymore, unless you choose to. All those "the web is the new operating system" people of the early 2000s finally won, we just forgot to celebrate.

But the liberation didn't last long. If you deploy software, you probably pay rent to AWS.

Why is that? Compute, right? AWS provides scalable computing resources.

Well, you'd think so. But lots of people sell computing resources way cheaper. Even a mid-range Macbook can do 10x or 100x more transactions per second on its SSD than a supposedly fast cloud local disk, because cloud providers sell that disk to 10 or 100 people at once while charging you full price. Why would you pay

exorbitant fees instead of hosting your mission-critical website on your super fast Macbook?

We all know why:

Location, location, location. You pay exorbitant rents to cloud providers for their computing power because your own computer isn't in the right place to be a decent server.

It's behind a firewall and a NAT and a dynamic IP address and probably an asymmetric network link that drops out just often enough to make you nervous.

You could fix the network link. You could reconfigure the firewall, and port forward through the NAT, I guess, and if you're lucky you could pay your ISP an exorbitant rate for a static IP, and maybe get a redundant Internet link, and I know some of my coworkers actually did do all that stuff on a rack in their garage. But it's all a lot of work, and requires expertise, and it's far away from building the stupid dashboard or blog or cat video website you wanted to build in the first place. It's so much easier to just pay a hosting provider who has all the IP addresses and network bandwidth money can buy.

And then, if you're going to pay someone, and you're a serious company, you'd better buy it from someone serious, because now you have to host your stuff on their equipment which means they have access to... everything, so you need to trust them not to misuse that access.

You know what, nobody ever got fired for buying AWS.

That's an IBM analogy. We used to say, nobody ever got fired for buying IBM. I doubt that's true anymore. Why not?

I refuse to say pendula.

IBM mainframes still exist, and they probably always will, but IBM used to be able to charge rent on every aspect of business computing, and now they can't. They started losing influence when

Microsoft arrived, stealing fire from the gods of centralized computing and bringing it back to individuals using comparatively tiny underpowered PCs on every desk, in every home, running Microsoft software.

I credit Microsoft with building the first widespread distributed computing systems, even though all the early networks were some variant of sneakernet.

I think we can agree that we're now in a post-Microsoft, web-first world. Neat. Is this world a centralized one like IBM, or a distributed one like Microsoft?

[When I did this as a talk, I took a poll: it was about 50\50]

So, bad news. The pendulum has swung back the other way. IBM was centralized, then Microsoft was distributed, and now the cloud+phone world is centralized again.

We've built a giant centralized computer system, with a few megaproviders in the middle, and a bunch of dumb terminals on our desks and in our pockets. The dumb terminals, even our smart watches, are all supercomputers by the standards of 20 years ago, if we used them that way. But they're not much better than a VT100. Turn off AWS, and they're all bricks.

It's easy to fool ourselves into thinking the overall system is distributed. Yes, we build fancy distributed consensus systems and our servers have multiple instances. But all that runs centrally on cloud providers.

This isn't new. IBM was doing multi-core computing and virtual machines back in the 1960s. It's the same thing over again now, just with 50 years of Moore's Law on top. We still have a big monopoly that gets to charge everyone rent because they're the gatekeeper over the only thing that really matters.

Sorry, just kidding.

Connectivity.

Everyone's attitude is still stuck in the 1990s, when operating systems mattered. That's how Microsoft stole the fire from IBM and ruled the world, because writing portable software was so hard that if you wanted to... interconnect... one program to another, if you wanted things to be compatible at all, you had to run them on the same computer, which meant you had to standardize the operating system, and that operating system was DOS, and then Windows.

The web undid that monopoly. Now javascript matters more than all the operating systems put together, and there's a new element that controls whether two programs can talk to each other: HTTPS. If you can HTTPS from one thing to another, you can interconnect. If you can't, forget it.

And HTTPS is fundamentally a centralized system. It has a client, and a server. A dumb terminal, and a thing that does the work. The server has a static IP address, a DNS name, a TLS certificate, and an open port. A client has none of those things. A server can keep doing whatever it wants if all the clients go away, but if the servers go away, a client does nothing.

We didn't get here on purpose, mostly. It was just path dependence. We had security problems and an IPv4 address shortage, so we added firewalls and NATs, so connections became one way from client machines to server machines, and so there was no point putting certificates on clients, and nowadays there are 10 different reasons a client can't be a server, and everyone is used to it, so we design everything around it. Dumb terminals and centralized servers.

Once that happened, of course some company popped up to own the center of the hub-and-spoke network. AWS does that center better than everyone else, fair and square. Someone had to. They won.

Okay, fast forward. We've spent the last 5 years making Tailscale the solution to that problem. Every device gets a cert. Every device gets an IP address and a DNS name and end-to-end encryption and an identity, and safely bypasses firewalls. Every device can be a peer. And we do it all without adding any latency or overhead.

That's the New Internet. We built it! It's the future, it's just unevenly distributed, so far. For people with Tailscale, we've already sliced out 10 layers of nonsense. That's why developers react so viscerally once they get it. Tailscale makes the Internet work how you thought the Internet worked, before you learned how the Internet works.

I like to use Taildrop as an example of what that makes possible. Taildrop is a little feature we spent a few months on back when we were tiny. We should spend more time polishing to make it even easier to use. But at its core, it's a demo app. As long as you have Tailscale already, Taildrop is just one HTTP PUT operation. The sender makes an HTTP request to the receiver, says "here's a file named X", and sends the file. That's it. It's the most obvious thing in the world. Why would you do it any other way?

Well, before Tailscale, you didn't have a choice. The receiver is another client device, not a server. So it was behind a firewall, with no open ports and no identity. Your only option was to upload the file to the cloud and then download it again, even if the sender and receiver are side by side on the same wifi. But that means you pay cloud fees for network egress, and storage, and the CPU time for running whatever server program is managing all that stuff. And if you upload the file and nobody downloads it, you need a rule for when to delete it from storage. And also you pay fees just in case to keep the server online, even when you're not using it at all. Also, cloud employees can theoretically access the file unless you encrypt it. But you can't encrypt it without exchanging encryption keys somehow between sender and recipient. And how does the receiver even know a file is there waiting to be received in the first place? Do we need a push notification system? For every client platform? And so on. Layers, and layers, and layers of gunk.

And all that gunk means rent to cloud providers. Transferring files — one of the first things people did on the Internet, for no extra charge, via FTP — now has to cost money, because somebody has got to pay that rent.

With Taildrop, it doesn't cost money. Not because we're generously draining our bank accounts to make file transfers free. It's because the cost overhead is gone altogether, because it's not built on the same devolved Internet everyone else has been using.

Taildrop is just an example, a trivial one, but it's an existence proof for a whole class of programs that can be 10x easier just because Tailscale exists.

The chain of dominoes starts with connectivity. Lack of connectivity is why we get centralization, and centralization is why we pay rent for every tiny little program we want to run and why everything is slow and tedious and complicated and hard to debug like an IBM batch job. And we're about to start those dominoes falling.

The glimpse at these possibilities is why our users get excited about Tailscale, more than they've ever been excited about some VPN or proxy, because there's something underneath our kind of VPN that you can't get anywhere else. We're removing layers, and layers, and layers of complexity, and making it easier to work on what you wanted to work on in the first place. Not everybody sees it yet, but they will. And when they do, they're going to be able to invent things we could never imagine in the old centralized world, just like the Windows era of distributed computing made things possible that were unthinkable on a mainframe.

But there's one catch. If we're going to untangle the hairball of connectivity, that connectivity has to apply to...

There's going to be a new world of haves and have-nots. Where in 1970 you had or didn't have a mainframe, and in 1995 you had or didn't have the Internet, and today you have or don't have a TLS cert, tomorrow you'll have or not have Tailscale. And if you don't, you won't be able to run apps that only work in a post-Tailscale world.

And if not enough people have Tailscale, nobody will build those apps. That's called a chicken-and-egg problem.

This is why our company strategy sounds so odd at first glance. It's why we spend so much effort giving Tailscale away for free, but also so much effort getting people to bring it to work, and so much effort doing tangential enterprise features so executives can easily roll it out to whole Fortune 500 companies.

The Internet is for everyone. You know, there were internetworks (lowercase) before the Internet (capitalized). They all lost,

because the Internet was the most diverse and inclusive of all. To the people building the Internet, nothing mattered but getting everyone connected. Adoption was slow at first, then fast, then really fast, and today, if I buy a wristwatch and it doesn't have an Internet link, it's broken.

We won't have built a New Internet if nerds at home can't play with it. Or nerds at universities. Or employees at enterprises. Or, you know, eventually every person everywhere.

There remain a lot of steps between here and there. But, let's save those details for another time. Meanwhile, how are we doing?

Well, about 1 in 20,000 people in the world uses the New Internet (that's Tailscale). We're not going to stop until it's all of them.

I'm old enough to remember when people made fun of Microsoft for their thing about putting a computer on every desk. Or when TCP/IP was an optional add-on you had to buy from a third party.

You know, all that was less than 30 years ago. I'm old, but come to think of it, I'm not that old. The tech world changes fast. It can change for the better. We're just getting started.