Remembering the LAN

2020-01-28

A memory and a dream.

How it was

I started programming in the 1990s living above my parent's medical practice. We had 15 PCs for the business, and one for me. The standard OS was MS-DOS. The network started off using IPX over coax to a Novell Netware server, the fanciest software we ever owned. IPX was so much easier than TCP/IP. No DHCP and address allocation, it just worked.

Eventually the PCs would run Windows, and a Windows NT server took over file sharing over TCP/IP. The business software survived this transition unchanged, though there was more operational overhead. We assigned IPs manually.

Home was a small town in Northern Australia. The internet was far off for me at this point, and would remain so longer than it did elsewhere. Eventually we would be able to make long-distance phone calls 2000 miles to try it out for a few minutes here and there. (At this point Americans had AOL.)

Before we had internet there were some lackluster local BBSs, and at one point a local university account my father acquired (somehow or other, none of us were students or university employees) that we could dial into and try out my first Unix on a Sun box. It was a limited experience even though technically it was on an internet link. My distance from university culture meant I wouldn't see Linux until the mid-90s, when we picked up a copy of Slackware on a trip to Hong Kong. I didn't really get Unix until I used OpenBSD, which put enough of the pieces together for me for Unix to finally make sense.

I wouldn't see root on a Sun box for more than a decade, now the early 2000s, when I bought half a dozen UltraSPARC servers as a lot second hand in Berkeley (for around $100, a lot of money for me at the time). I assembled a working machine from the carcasses and used it to write a sparc64 C compiler backend. Even though more time has passed since than between these times, it is hard for me to hold both lives in my head simultaneously. They were different worlds.

The childhood magic

The LAN was a magical place to learn about computers. Besides the physical aspect of assembling and disassembling machines, I could safely do things unthinkable on the modern internet: permission-less file sharing, experimental servers with no security, shared software where any one machine could easily bring down the network by typing in an innocuous command. Even when I did bring down the network the impact never left the building. I knew who I had to apologise to.

(Two decades later when I took down a borgmaster with a misconfigured MapReduce as an engineer at Google, I could not figure out who should get an apology email.)

With our LAN easy things were easy, and some hard things were possible. There were high-level interpreted languages where UIs were straightforward, and scary languages which could make the computer really shine.

A 200MHz Pentium Pro felt blazing fast and 32MB of RAM could do anything. By the time I had OpenBSD I could recompile Apache httpd in a few minutes with my own bad ideas. My wrist watch could compile it faster today, as long as I stuck to GCC 2.95.

Later I would carry a PC to houses of my friends where we would build ephemeral LANs and play games like Starcraft. (Cathode-ray monitors were heavy.) The LAN was an education and a lifestyle.

The small business magic

My father, a general practitioner, used this infrastructure of cheap 286s, 386s, and 486s (with three expensive laser printers) to write the medical record software for the business. It was used by a dozen doctors, a nurse, and receptionist. You can do a lot with file-based database software (in this case, Clipper) and a mouse-less curses interface.

There are several astonishing facts about this. As an engineer, it is astonishing that Netware file locking, then SMB file locking, worked well enough to implement a database used by ~15 concurrent users. I suspect most career programmers today have never used file locking, let alone seen it work correctly.

The business story is even more astonishing. Here is a non-programming professional, who was able to build the software to run their small business in between shifts at their day job using skills learned from a book.

Today a professional could surely pick up the skills to build a CRUD app, but they would be hard pressed to tune their software so relentlessly to minimize the keystrokes a receptionist needs to use to check-in a patient, or support a magnetic card reader, or teach laser printers to precisely print onto specialized prescription paper (the printers spoke postscript, but the MS-DOS programming language had an easier instruction layer over PS that made this possible).

The result in the 90s was the business needed fewer staff than everyone thought a medical practice of that size required, doctor's time was used more efficiently than any other software allowed, so productivity increased.

My father made more money as a part-time programmer optimizing his small business than he did as a doctor seeing patients.

How it is

If my 90s childhood were transported to today, so many new things would be possible. I could draw high-quality graphics easily with JavaScript. It is not clear that would be more compelling than the pixelated gorillas and bananas I played with in BASIC. I could develop apps for my phone. In theory at least. In practice, I wasn't particularly patient with slow compilers as a kid, and as an adult I still have trouble stomaching the development environment for apps, so that's off the table.

I wouldn't build a toy website to put school stuff on, because I would have facebook for that.

Games would be easier to play with friends. We wouldn't have to lug heavy boxes or learn to debug our TCP/IP configuration or actually see each other in person to play. I guess some people would see that as an improvement: more candy, less content.

All the technology is better. The resources to learn are better. But it is not clear to me I would program at all today. Learning how to store passwords or add OAuth2 to your toy web site is not fun. So much of programming today is busywork, or playing defense against a raging internet. You can do so much more, but the activation energy reqired to start writing fun collaborative software is so much higher you end up using some half-baked SaaS instead.

What about my father?

Could a part-time programmer like my father write small-business software today? Could he make it as safe and productive as our LAN was? Maybe. If he was canny, and stuck to old-fashioned desktops of the 90s and physically isolated the machines from the internet. But there is no chance you could get the records onto a modern phone safely (or even legally under HIPPA) with the hours my father gave the project.

If confronted with the build v. buy decision today, I strongly suspect he would buy. Or even more likely, subscribe. The practice would be less productive for it.

The programmers of the world have built this fantastic internet, full of magic. Free inter-continental video calls. "Micro" VMs available for free from Cloud providers with more processing power and memory than anything I could have bought when I started programming.

For all our mastery, something has been lost. If programming a LAN in the 1990s was the care-free tending to a garden in the countryside, then programming on the internet today is tending a planter box on Madison Avenue in midtown. Anyone can experience your work. You will also have your tilling judged by thousands of passersby, any of whom may ruin your work because the dog they're walking hasn't been city trained.

A dream: How it will be

In some moments the right threads of change meet and create something special. Many of these moments are short-lived and will not repeat, destined to be, at best, remembered.

The magic moment of small trusted networks and care-free programs does not need be relegated to memory. With enough work, we can bend technology to recreate the magic.

We can have the LAN-like experience of the 90's back again, and we can add the best parts of the 21st century internet. A safe small space of people we trust, where we can program away from the prying eyes of the multi-billion-person internet. Where the outright villainous will be kept at bay by good identity services and good crypto.

The broader concept of virtualizing networks has existed forever: the Virtual Private Network. New protocols make VPNs better than before, Wireguard is pioneering easy and efficient tunneling between peers. Marry the VPN to identity, and make it work anywhere, and you can have a virtual 90s-style LAN made up of all your 21st century devices. Let the internet be the dumb pipe, let your endpoints determine who they will talk to based on the person at the other end.

The result is a system with properties that work with today's internet to give us the pleasant, simple programming environment of the '90s LAN:

- Use the global internet identity system of your choice for authentication, and do cryptographic authorization at the IP level.
- Keys are generated and rotated for you automatically.
- People map directly to unspoofable IP addresses.
- Run custom servers on your network and access is limited to only those people on the network.
- Your data is protected by the simple yet powerful social dynamics of small groups.

We can build this.

First, we have to prove that the user experience creates the environment we want for simpler programming. That means getting the product in the hands of customers and making them happy. This is our current focus at Tailscale, build a great product and make customers happy.

Second, we need to stabilize and publish the protocols used to build this mesh overlay network so that it can be used anywhere.

Third, I need to help new programmers who never got to experience simple, pleasurable programming in a safe environment understand that programming can be fun. You can set up your environment so you can focus on being creative. Writing a web service for use by your friends should not be a form of combat, where you spend your days worrying about XSS attacks or buffer overflows. You should be focused on creating something new and wonderful in a place without bad people hounding you.

We are going to rebuild the LANs (and BBSs and MUDs) of the 90s as a world of mesh networks on top of today's internet.

Index
github.com/crawshaw
twitter.com/davidcrawshaw
david@zentus.com