The NCP address crunch

Posted on 1 April 2011 by meta

http://meta.ath0.com/2011/04/01/the-ncp-address-crunch/

[Note: This text was posted through a time warp from 1979. Special thanks to Dan Bernstein and Avery Pennarun for their technical expertise.]

Computers on the ARPAnet talk to each other using NCP, the Network Control Protocol. Each computer on the ARPAnet has its own public NCP address, similar to a street number; for example, 207. The target of each packet of data is identified by a public NCP address.

Problem: There are only 256 NCP addresses. Many of those addresses have already been allocated. What happens when we run out of public NCP addresses?

Partial solution: Do all these computers really need to be on the ARPAnet? A company with 20 computers browsing FTP sites doesn't need to put all those computers on the ARPANET. It can have a single computer on the ARPAnet (a "proxy") that retrieves data from FTP servers on behalf of the other 19 computers, forwards telephone-over-NCP calls from those computers, etc.

Most people agree, however, that proxies merely delay the inevitable.

Long term solution: IPv4, the new Internet protocol, has many more addresses.

Basic interoperability issues

Suppose someone sells you a public IPv4 address. You put your computer on that address. You find that you can't reach your company's FTP servers. How will you react?

This is an example of what's called an **interoperability failure**. Right now, many–in fact, most–network services can't talk to clients on public IPv4 addresses. Until this changes, using a public IPv4 address instead of an NCP address will be a disaster for servers.

The IPv4 designers made a fundamental conceptual mistake: they designed the IPv4 address space as an alternative to the NCP address space, rather than an extension to the NCP address space.

This might sound like a very small mistake: after all, once IPv4 is working, we can move everything to IPv4, so who cares about NCP? The problem is that this mistake has gigantic effects on the cost of making IPv4 work in the first place.

Each of the server admins must acquire his own public IPv4 address, announce that alongside his NCP address, and configure the server to respond to that address alongside its public NCP address, or else the client will fail.

If I run an IPv4-only server, people with NCP can't connect to it, and at least one valuable person is surely going to remain on NCP. So if I adopt IPv4, I adopt it in addition to NCP, not in exclusion. IPv4 will only start to be useful once I can turn off NCP, which will only be after all servers have IPv4 addresses. Any transition plan involves everyone having an IPv4 address, and until then we can't start dropping NCP, and IPv4 won't be useful. This is a classic chicken-and-egg problem, and it's unsolvable by brute force. *

But really, there's only one thing that makes IPv4 undesirable, but it's a doozy: **the addresses are just too annoyingly long**. I can easily remember that DEC is 79, Xerox PARC is 32, and BBN is 72. With the new IPv4, I'm expected to memorize 15.216.110.22 to reach HP, 13.7.8.141 to reach PARC, and 192.1.98.3 to reach BBN. Instead of three simple numbers, I now have to memorize four times as many, *and* make sure I get them in the right order. Madness.

So IPv4 addresses are impossible to memorize. What's more, auto-renumbering of hosts via DHCP means that anything I memorize today might be totally different tomorrow. IPv4 addresses are like GUIDs. If they were a good idea, people would use them instead of URLs. Are URLs perfect? Does anyone love Network Solutions? No, of course not. But it's 1000x better than looking at http://97.42.88.4/ and trying to guess if that's *really* my bank's web site or not.

The counterargument, of course, is that DNS is supposed to solve this problem. Give each host a ~~GUID~~ IPv4 address, and then just map a name to that address, and you can have the best of both worlds.

But DNS is a steaming pile of hopeless garbage. When I bring my laptop to my friend's house and join his LAN, why can't I see the other hosts on his LAN by name? Because DNS sucks. I'd rather they fixed that problem today before making me switch to something where I can't possibly remember my host address.

Basically, we need some sort of DNS protocol where new hosts on a network can exchange name information with zero configuration required. And a way for servers to multicast DNS information to allow some sort of service discovery. While they're at it, they should find a way for IP addresses to be assigned automatically without needing a DHCP server to be set up. If they did all that, it might make IP usable. Perhaps eventually they might even make printers and other devices work with this magical zero configuration DNS.

But while we wait for the impossible, there's a solution to this whole IPv4 mess right now. We just need to skip the whole boondoggle. As already mentioned, not everyone in the world needs a public NCP address. In fact, most people don't need one, because most people make only outgoing connections. Their internal network address can be mapped to the external address by their IMP router, via Network Address Translation (NAT).

OK, you say, but don't all servers need an NCP address? Not at all! HTTP/1.1, which is what everyone uses now, supports "virtual hosts". You can connect to an NCP address on port 80, and you provide a Host: header at the beginning of the connection, telling it which server name you're looking for. The NCP address can then decide to route that request anywhere it wants.

So your request for www.google.com at address 79 could be routed by IMP to a different address, say 15, on BBN's internal network. That server would look at the header and route via an internal connection to Google's IMP, which might be address 32 on the internal BBN network. So no, we are not going to run out of NCP addresses, because the web is all anyone uses any more and HTTP is infinitely proxyable.