Network Working Group
Request for Comments: 1681
Category: Informational

S. Bellovin
AT&T Bell Laboratories

August 1994

On Many Addresses per Host

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document was submitted to the IETF IPng area in response to RFC 1550. Publication of this document does not imply acceptance by the IPng area of any ideas expressed within. Comments should be submitted to the big-internet@munnari.oz.au mailing list.

Overview and Rational

Currently, most hosts have only one address. With comparatively rare exceptions, hosts as hosts -- as opposed to hosts acting as routers or PPP servers -- are single-homed. Our address space calculations reflect this; we are assuming that we can estimate the size of the address space by counting hosts. But this may be a serious error. I suggest that that model may -- and should -- change.

For the ideas outlined below, I do not claim that multiple addresses per host is the only or even necessarily the best way to accomplish the goal. I do claim that my ideas are at the very least plausible, and that I expect that many of them will be tried.

Encoding Services

More and more often, services are being encoded in the host name. One can fetch files from ftp.research.att.com, look up an IP address on ns.uu.net, synchronize clocks from ntp.udel.edu, etc. Should this practice be generalized to the IP address domain?

In some cases it would be a very good idea. Certain services need to be configured by IP address; they are either used when the DNS is being bootstrapped (such as in glue records and root server cache records), or when its unavailable (i.e., when booting after a power hit, and the local name servers are slower to reboot than their diskless clients.

Security is another reason, in some cases. Address-based authentication is bad enough; relying on the name service adds another layer of risk. An attacker can go after the DNS, in that case. A risk-averse system manager might prefer to avoid the extra exposure, instead granting privileges (i.e., rlogin or NFS) by address instead of name. But that, of course, leads to all the usual headaches when the location of the service changes. If the address for the service could be held constant, there would be much more freedom to move it to another machine. One way to do that is by assigning the serving host a secondary address.

A related notion comes from the need to offer different views of a service from a single host. For example, research.att.com has long offered two distinct FTP archives, with slightly different access policies. It would be nice if both could live on the same machine, without asking the user community to learn new protocols or custom port numbers.

Archie is an even better example. There are three principal ways to use Archie: use a special protocol, and hence a special application program, on a dedicated port and host that is probably named archie.foo.bar; telnet to archie.foo.bar and go through an extra and gratuitous login as archie, or telnet to some special port on archie.foo.bar. The latter two are examples of using a standard protocol (telnet) to offer a different service. Neither alternative is very convenient.

It would be better if archie.foo.bar provided the Archie service, while host.foo.bar provided a login prompt. Again -- an easy way to do this is to assign the host a separate IP address for its extra service.

Note that there are security advantages here, too. A firewall could be configured to allow access to the address associated with the Archie server, but not the other addresses on that host. That would provide a high degree of safety, assuming, of course, that the other servers on that host were bound to its primary addresses, and not the exposed address.

Another way to implement this concept would be to extend the DNS, to return port number information as well as IP addresses. Thus, netlib.att.com might return 192.20.225.3/221. But that would necessitate changing every FTP client program, a daunting task.

We could also look on this as the extension of the MX concept. MX records are very valuable, but they apply only to mail, and they don’t supply port numbers. Again, changing this would require massive client program changes.

Accounting and Billing

For better or worse, some parts of the Internet are moving towards usage-sensitive charging. At least four charging schemes seem possible; doubtless, the marketeers in charge of such things can and will come up with more.

The first is the traditional "pay as you go" approach. Each host is responsible for its own packets. Of course, that means that in a typical conversation, both parties pay -- and the providers of free FTP archives will end up paying dearly for their beneficence. That leads to our second model: caller pays. Other people might want to make collect calls, much as is done on the telephone today. Finally, there might be the equivalent of American "900" numbers: the caller pays a premium to the server.

This is not at all far-fetched; UUNET already has a 900 number for anonymous uucp clients. No need to register in advance; just dial in, and let the phone company act as your agent.

Given all these schemes, it is vital that the caller and recipient know in advance who will pay. It is not acceptable for users to learn, only after the fact, that they have incurred a cost. We could envision use of IP options, but again, that would preclude use of today's standard clients.

It is not sufficient to present a message at connection time warning of the charges. Many interactions do not provide a hook for user interaction. And there are security concerns -- suppose that someone puts up a gopher server that redirects a caller to some pay-to-play address, without displaying the required warning. A scam? Sure -- but it's already happened with the phone network, and I see no reason to think that the Internet will be far behind.

My suggestion, of course, is to encode the charge algorithm in the destination address (and perhaps in the DNS name space as well). The bits themselves would determine who pays. Organizational border routers could implement policies on pay services; the anonymous workstations in a dorm computer lab wouldn't be allowed to call collect.

An extension of this scheme would use a comparatively large number of bits, letting the address act not just as a policy indicator, but also as an index to a charge algorithm table.

Addresses per User

It may be useful to assign each user on a host a separate IP address, for the duration of the login session. This has a number of advantages.

The first ties in with the charging scheme given above. Usage-sensitive accounting today is done by routers, and they have no notion of who is using the hosts. If each user had a separate IP address, we could continue to gather the accounting data at the router. The host would simply have to record the address assignments; billing could be done offline.

Similarly, different classes of users could have different forms of addresses. Those with hard-money accounts might have some bits set in the address that would allow for access to costly services. The border routers could make this sort of distinction, using today's technology.

An IP address per user also fits in well with encryption. There is a lot of attention today focused on network-layer encryption. But that provides host-level granularity of protection, which is sometimes insufficient. Transport-layer encryptors provide finer-grained protection, but does the Internet need two different low-level encryption schemes? If each user had a separate IP address -- and perhaps had it only on hosts that cared about such matters -- we could provide user-level protection and accounability, with the same infrastructure used to support host-level accountability.

Low-Grade Mobility

There are several schemes under discussion for mobile IP hosts. These are aimed at a fairly general model of hosts moving anywhere. While that is important, there is also some need for limited mobility, within a subnet. This could be used for load-balancing. A mail relay that had just been asked to send a large message to a huge mailing list could offload some of its IP addresses to its peers. That would divert future incoming messages without invalidating thousands of cached MX records and their associated IP addresses. Similarly, servers for low-speed X terminals could reside on different physical machines, all the while not disturbing sessions in progress.

Merging Subnets

There has long been some need to merge subnets. Sometimes this is due to organizational changes; other times, people have installed bridges when routers would have been a more appropriate choice. Some

hosts need to live on both logical networks at once, to avoid an extra hop through a router. It would be useful to be able to assign them such addresses.

How Many Addresses Do We Need?

Assuming that some of these ideas bear fruit, how many addresses do we need, per host?

Most of these schemes are fairly cheap. Few people would offer more than a handful of distinct service views per system. But the address-per-user notion could be quite costly. We also have to account for address mask assignment policies. In many of today's networks, enough bits of host address have to be allocated to allow for the largest subnet in an organization. Even if we assume that IPng's routing protocols will be smarter about such things, foresight in address allocation will be needed to allow headroom for some networks to grow, while still maintaining a contiguous netmask. This in turn will contribute to sparse utilization of the address space. Accordingly, I recommend that we allow for 2^6, and perhaps as many as 2^8, extra addresses per host, to leave room for the ideas presented here.

I should note that the idea of encoding the service in the transport address bears some relation to OSI's model. That similarity should not, of course, invalidate the idea.

Acknowledgements

Some of these ideas were derived from conversations with Matt Blaze.

Security Considerations

Security issues are discussed throughout this memo.

Author's Address

Steven M. Bellovin
Software Engineering Research Department
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA

Phone: +1 908-582-5886
Fax: +1 908-582-3063
EMail: smb@research.att.com