London Communication Symposium (LCS), University College London, London, UK. Sep 2004.

Naming Enhancements for the Internet

R. Atkinson & S. Bhatti
Department of Computer Science
University College London

August 17, 2004

Abstract

The Internet has a number of different namespaces used by its myriad protocols and applications. Unfortunately overloaded naming semantics are creating widespread issues. This position paper reviews the existing namespaces, current naming issues, and then proposes a strawman approach to resolving the current naming issues.

1 Existing Namespaces

In the original ARPAnet, the first namespace was the address. The modern Internet primarily uses a 32-bit IPv4 address, for example 128.16.6.8, and experimentally is using the 128-bit IPv6 address. The network-layer address was originally intended for the routing of packets, but currently is misused as a host identifier by upper-layer protocols. Closely associated with the IP address is the IP Subnetwork name, which consists of a routing prefix (up to 32 bits) and a prefix length (ranging from 1 bit to 32 bits) in combination. The IP Subnetwork name is primarily used within routing protocols and the routing system.

The upper-layers of the Internet Architecture have additional namespaces. Working our way up the stack, we first encounter the Connection End Point. This is the triple consisting of a host, represented as an IP Address or domain name, a transport-protocol, for example the Transmission Control Protocol (TCP), and a Port number, for example port 25 which is used for electronic mail sent via the Simple Mail Transfer Protocol (SMTP).

As we approach the application-layer, we encounter the Domain name, for example *cs.ucl.ac.uk*, and its derivative the Mailbox name, for example *A.Lastname@cs.ucl.ac.uk*. The Mailbox normally names a specific user, though in some cases it names a set of users, for example when the Mailbox names a mailing-list. Alternately, the mailbox might provide a role-based name, for example *(postmaster,webmaster)@cs.ucl.ac.uk*. So we have overloaded semantics for a mailbox name, which might refer explicitly to a particular human user, other times to a set of users (e.g. mailing list), and yet other times with implicit semantics of a service name (e.g. the user(s) act as the electronic postmaster or webmaster for UCL's Department of Computer Science).

Most recently, the *Universal Resource Locator (URL)*, and its derivatives the *Universal Resource Name (URN)* and *Universal Resource Identifier (URI)* have appeared. By now the URL is familiar to nearly everyone, as it is the name used to access web content. If one wanted to read the latest news from cable-TV network CNN, one would use the URL *http://www.cnn.com*. In this, the first portion, *http*, names an invocation method, such as the Hyper-Text Transfer Protocol (HTTP), then comes a Domain name (www.cnn.com), and then optionally the name of a specific object associated with that Domain name. The URI and URN differ from the URL in that they name objects without

necessarily indicating how to locate the object on the network. In the example above, the domain name has the implicit semantics of a service name. Hence, one can see that we have overloaded semantics, whereby a domain name might sometimes refer to a host, other times to a domain (set of hosts), and yet other times to a service for the specified domain.

2 A Brief History of Naming in the Internet

The network-layer address is the most fundamental namespace in a packet network and necessarily existed from the start. However, even early ARPAnet users quickly realised that a more human-friendly name was needed for hosts. This was originally implemented as a flat text file named *HOSTS.TXT*[Kud74]. At this time there was no concept of a heirarchical Domain name. So, for example, the UCL host attached to the ARPAnet might have been named UCL-SATNET.

In the early 1980s, the University of California at Berkeley were developing the Berkeley Software Distribution (BSD) of the AT&T Unix operating system. For the 4.2 BSD release, UCB ported a TCP/IPv4 networking stack into Unix and of course also added the *Sockets API* to BSD Unix. Because there was no Domain Name System (DNS) at the time, UCB's *Sockets API* uses raw IP Addresses rather than supporting Domain names.

The Domain Name System (DNS) was deployed in the late 1980s and enhanced the previous host name system in at least two major ways[MD88]. Domain names are explicitly hierarchical, which simplified name administration. Also, the DNS included a protocol used for resolving Domain names into addresses. Since its original deployment, the DNS has been extended with service location capabilities, for example using the MX, KX, or SRV resource records. In modern usage, a Domain name might be used to name a single host, an administrative domain, a service, or even a cluster of servers that are masquerading as a single host. These overloaded semantics make the DNS more complex, and in some ways less useful.

Early on, IPv4 addresses had an implicit network mask; by examining the first octet, one could determine which of three classes the network was in and hence which network mask (either 8, 16, or 24 bits) was associated with the address. This inflexibility, combined with ad-hoc address allocation practices led to excessive routing table growth in the early 1990s. This was fixed by the combination of class-less routing, where a variable-length network mask was explicitly specified, and more systematic address allocation policies. For many reasons, including simplified address management and perceived security advantages, Network Address Translation (NAT) has come into common use.

3 Current Naming Issues

At first glance, it might appear that the Internet already has a relatively rich set of namespaces. However, there are several significant current issues. The implicit overloading of Domain names to names of services, rather than names of hosts, for example *www.cnn.com*, creates issues. Also, the current scheme is unable to support multiple instances of a given service (e.g. on different TCP ports) on a single host. **There is a clear need to create explicit service names.**

Also, for historical reasons, most networking APIs lack appropriate abstractions, instead forcing application programmers to use inappropriately low-level objects. Also, if proper naming abstractions were in use, applications would not be adversely impacted if a host's address changed as a result of the host

moving to a new location. Instead, many applications and most transport-layer protocols contain the IP address in local state.

A Domain name might be a host, a cluster, or the single interface of a host. Absent other information, one can't distinguish precisely what is being named for a given name. Similarly, an IP address actually names a single routing interface on a host, rather than the host itself. The misuse of the IP address as a generic host identifier, rather than a routing interface identifier, makes host mobility difficult and causes many applications to be incompatible with NAT.

4 Proposed Approach

Our proposal adds explicit Service names and generic network-layer Host Identifiers. In our scheme, IP Addresses resume their original limited role, are used only by the network-layer and only for packet routing. Indeed, this is in keeping with the current strong binding between an IP address and an interface. Transport protocols and Application protocols substitute more appropriate identifiers (e.g. the new Host Identifier or perhaps a Domain name) replacing current inappropriate uses of the IP Address. There still might be a few special instances (e.g. control messages, network management) where one would still want to use raw IP addresses. Finally, the networking APIs are replaced with a new set of APIs that use more appropriate object types, thereby encouraging application programmers to use proper abstractions.

A candidate instantiation of this adds a new Identifier (ID) resource record to the DNS, with support for a mapping from a Domain name to the ID record. Of course, one can also use the existing PTR record to obtain the Domain name associated with a given IP Address, and thus traverse the DNS from Address to ID. There are existing specifications for DNS Security and for Secure Dynamic DNS Update, which provide authentication for these various name bindings [3rd99][Wel00]. For example, when a host moves, it can update its address records at its DNS server using the Secure Dynamic DNS Update. If this new identifier in the ID record contains suitable structure, then an authenticated DNS lookup can provide the domain-name associated with that ID.

Additionally, we propose extensions to the existing Internet Control Message Protocol (ICMP) to support host mobility. For example, *Host Has Moved* or *My New Address Is* ICMP messages are created to enable a mobile host to inform existing correspondent hosts of the new location of the mobile host. One can use IP Security to authenticate these messages and prevent forgeries.

Finally, existing transport-layer protocols (e.g. TCP) and application-layer protocols (e.g. FTP) are modified to replace current use of addresses with new use of more appropriately abstract identifiers. We propose a new service instance location protocol that is used to find the location of the service instance requested by the application via the new API.

There are a number of details that remain to be sorted out, in altering existing protocols, in properly designing and specifying the new mechanisms, and in providing some transition scheme from current systems to the conceptual new system.

5 Benefits

This new enhanced naming architecture provides a number of benefits in routing, in security, and in other areas. The routing system benefits from two improvements. First, host mobility is much simpler than the current *Mobile IP*or *Mobile IPv6* specifications permit. Second, the inter-domain routing system does not

suffer adverse impacts from widespread campus-level multi-homing. Each of these benefits derives primarily from the decoupling of the IP address used by the routing system from upper-layer protocols' need for some form of generic host identifier. That change permits in-transit rewriting of addresses without suffering adverse impacts at the upper protocol layers.

From a security perspective, the current practice of assuming that an IP address is a de-facto self-authenticating identifier is wrong; forging IP addresses in packets is trivial. There are numerous known cases of deliberate packet misrouting to facilitate eavesdropping, and potentially modification, of in-transit packets. To remedy this, the new system uses readily authenticatible identifiers. This facilitates the deployment of existing security specifications, for example IP Security or routing protocol authentication. In the case of IP Security, one can now use ESP or AH through a NAT device or other circumstance where an address was modified in transit, without suffering a security vulnerability. Also, the use of IP Security with mobile hosts is now quite straight forward.

Adding explicit service names, which is another idea, simplifies network-wide service location. The current *Service Location Protocol (SLP)* is useful only within a single LAN segment; explicit service names (beyond the existing DNS SRV record) can facilitate finding services anywhere on the global Internet. Also, the new approach better supports situations when there is more than one instance of a given service on a given host. By reducing semantic overloading of namespaces and updating APIs to use more appropriate object types, application programming should be simpler and easier.

6 Future Work

This is work in progress at an early stage, and the aim is to build a cleaner use of names and identity, simplifying the IP architecture and DNS usage. Whilst we have exposed potential benefits and proposed a strawman mechanism for enhancing the DNS, further work is clearly required. Issues of APIs and backwards compatibility with current mechanisms is a key practical issue. For our proposal, we have yet to fully resolve how it would impact on (or be utilised for) multicast/anycast, what the security and possible denial of service (DoS) implications might be, and of course how the unique identifiers would be managed (creation, storage, distribution, and verification/authentication). Also there is related work within the wider research community that should be examined [IRT]. Additionally, we must consider the privacy implications of creating and using a unique host identifier.

References

[3rd99] D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535, Internet Society, March 1999.

[IRT] IRTF Host Identity Protocol Research Group. http://www.irtf.org.

[Kud74] M. D. Kudlick. Host names on-line. RFC 608, Internet Society, January 1974.

[MD88] Paul V. Mockapetris and Kevin J. Dunlap. Design of the Domain Name System. *ACM Computer Communication Review*, 18(4):123–133, August 1988.

[Wel00] B. Wellington. Secure Domain Name System Dynamic Update. RFC 3007, Internet Society, Nov 2000.