Thoughts on a New Namespace

R. Atkinson
Department of Computer Science
University College London

23 November 2004

Abstract

This working paper describes a possible design for a new Internet namespace. A primary goal of creating this new namespace is to use the new namespace as a non-topological network-layer identifier for hosts. By doing so, a number of current routing issues, particularly those related to mobile hosts and to multi-homed campuses become simpler to solve. As this is a working paper, there is also discussion of open issues with this design.

1 Introduction

At present, the Internet Architecture does not have a transport-layer identifier or a network-layer identifier that is independent of the host's Internet Protocol address (IP Address). Instead, the Internet protocol suite overloads the IP address to sometimes be a topologically significant address of a particular network interface of a host, while at other times trying to use the IP address as a generic network-layer identifier (i.e. outside the context of the routing system). A consequence of this is that many protocols outside the routing system contain source and/or destination IP addresses in their protocol state. For example, the User Datagram Protocol (UDP) and the Transport Control Protocol (TCP) include both source IP address and destination IP address inside the transport-layer state of a given session. [1]

2 Background

2.1 Early History

If one looks back at the history of the Internet, it becomes clear that the technology was built bottom-up, rather than top-down. In the late 1960s, the focus of attention was creating a workable packet network protocol,

[1] In BSD Unix terms, the IP addresses are part of the *Transport Control Block*.

the Network Control Protocol (NCP) [NKPC70], and deploying a working wide-area packet network. Hence, in those early days the application software necessarily used the network-layer address in many different ways. Also, mobile networks and highly mobile hosts were not part of the original ARPAnet, so it was quite reasonable to assume that a host's network-layer address would not change very often or very quickly. In such an environment, the use of the network-layer address as the principal host identifier would only be natural.[2] Also, at this time the current notion of a layered protocol architecture was not fully refined.

However, people could more easily remember a name than a number, so early naming was created in the early 1970s. These initially used a fixed flat-file host table, which quickly came under centralised management [Wat71]. At this time there were no domain-names. Instead, there were single string hostnames, so most sites included the site nickname in each host's name. For example, some host named "A" at the Information Sciences Institute might be named "ISI-A". During the 1970s, work on protocol architectures progressed, leading to a more clearly layered architecture, and ultimately into the creation of TCP, UDP, and IPv4 as distinct protocols with distinct functions. The flag-day transition from the NCP-based ARPAnet to the IPv4-based ARPAnet occurred on 1 January 1983.[Pos81a]

2.2 Influence of BSD UNIX

In the early 1980s, the Computer Systems Research Group (CSRG) at UC Berkeley added TCP/IP networking support into the Berkeley Software DIstribution (BSD) of UNIX.[LMKQ89] At that same time, the Domain Name System (DNS) of today was being designed.[Moc83] So once again, it was only natural to use the network-layer address as the connection identifier in the BSD Sockets Application Programming Interface (Sockets API). As networked applications grew during the last two decades, many applications unconsciously used the network-layer address as a host identifier. In the early period, there was no other option. However, in the late 1980s, the Domain Name System (DNS) was widely deployed.[Moc87] A key decision was to initially deploy the BIND DNS resolver as a library in application-space, rather than updating the BSD Sockets API to support the domain-name directly. This early design choice means that even today applications using the BSD Sockets API must internally convert a domain-name provided by the user into an IP address that can be supplied to the BSD Sockets API to open, alter, or close a network session. So networked applications are necessarily aware of the IP addresses associated with a network session and many such applications continue to use the IP address as a host identifier. Had the DNS been deployed in conjunction with a revised BSD Sockets API, then

[2]The term *socket* actually dates back to the early 1970s and referred to a communications endpoint in the NCP-based ARPAnet.[HHM71]

applications might well have used the domain-name as the host-identifier. In that case, the deployed Internet might well be much more compatible with network-address translation and also might have better support for mobile hosts and mobile networks.

2.3 Today's Internet

So for reasons that are almost entirely historical, the deployed Internet pervasively uses the network-layer address as a host identifier, even though the address is ill-suited for that use and the domain-name exists as a much better alternative for that purpose. This misuse creates a number of architectural and practical problems in the deployed Internet of today. The next section will discuss the current architectural issues created by the misuse of the address as a host identifier in more detail.

3 Current Architectural Issues

The existing Internet Architecture has performed admirably for many years. However, during the 1990s, some issues arose that could not have been foreseen and external business stresses also arose. As a result of this, there are some current architectural issues within the Internet Architecture. Among these issues are clean support for mobile hosts, scalable support for campus-network multi-homing, and the widespread deployment of Network Address Translation techniques. This section describes those issues in more detail and highlights a possible architectural change that might significantly ameliorate each of them.

A recurring issue is that the transport-layer state in the hosts contains network-layer address information. For many years, the Internet has had two primary transport-layer protocols, the User Datagram Protocol (UDP) [Pos81c] and the Transmission Control Protocol (TCP) [Pos81b]. UDP provides an unreliable datagram service, while TCP provides a byte-oriented reliable transport service. UDP can support unicast or multicast sessions, while TCP can only handle unicast sessions. The 4.x BSD implementation of UDP and TCP keeps transport-layer session state in the *Protocol Control Block* data structure. This state includes the IP addresses of the end points of the session, making it difficult to have any IP address change during the lifetime of the session. [LMKQ89][MBKQ96]

In recent years, the IETF has standardised a new transport-layer protocol, the Stream Control Transmission Protocol (SCTP).[OY02]. SCTP is similar to the Transmission Control Protocol (TCP) in some ways; each provides reliable in-order delivery of data, unlike UDP. However, SCTP provides message-oriented delivery rather than byte-oriented delivery. Also, SCTP explicitly supports multi-homed endpoints, although it does not support

mobile endpoints. Common implementations of SCTP also keep transport-layer session state in the Protocol Control Block data structure. While SCTP supports multi-homed endpoints, it only supports unicast sessions. So the Transport Control Block for SCTP holds more than one IP address for each endpoint, but does not support changing any IP address during a given SCTP session. So none of the standard transport-layer protocols have support for mobile hosts or for changes in any IP address during the lifetime of a given transport session. Further, the 4.x BSD implementation of those protocols does not support changes to the IP address(es) in the Protocol Control Block.

3.1 Mobile Hosts

If one considers the case of a laptop that is moving and also in use, there are at least two kinds of mobility. The first kind, which here we will call *micro-mobility*, includes mobile hosts that are able to retain link-layer connectivity while mobile. This might be implemented by a link-layer interface to a mobile phone system's data service or via a link-layer interface to a wireless Ethernet subnet, for example. The second kind, which here we will call *macro-mobility*, includes mobile hosts that are not able to retain link-layer connectivity while moving. In this paper, we will only be discussing macro-mobility. It is important to understand that since an IP Address names the location in the routing system of a particular network interface on a host, the IP Address will normally need to change whenever a host connects to a different link-layer segment (IP subnetwork).

A significant challenge in the current Internet Architecture is how a mobile host can retain an active TCP or UDP session even as its IP address needs to change to reflect a new location in the network's topology. This is challenging for the reason hinted at above, namely the transport-layer contains the IP addresses of the session endpoints in the transport-layer session state for those sessions. Also, various upper-layer protocols include IP addresses as identifiers in their session state, and sometimes even as protocol data elements over the wire.

One can imagine a variety of different 'hacks' that could be used to work around this architectural limitation, some more ugly than others, but none of them really attractive. In Mobile IP, the host retains a fixed *Home Address* permanently and the transport-layer session state includes that IP address, rather than the host's variable *Care-Of Address*. [Per02] There are additional protocol mechanisms that are used to enable the mobile host to inform its *Home Agent* about the mobile host's current Care-Of Address and to inform existing *Correspondent Hosts* whenever the Care-Of Address changes. Aside from the tremendous complexity that all these new protocol mechanisms bring, there are a variety of subtle security issues lurking in this approach.

If one had a non-topological network-layer *Identifier* in addition to the existing IP Address, then one could use that identifier in the transport-layer session state. Since the identifier is non-topological, its value would not change as the mobile host changed location. As a consequence of that, such an identifier could be used in the transport-layer state to decouple the transport-layer from any need to know about the location change of the network-layer interface. Similarly, upper-layer protocols that currently use the IP address in the upper-layer protocol state could instead use this new topology-independent identifer.

3.2 Multi-Homed Sites

It is no longer unusual for a campus network to have more than one upstream network link, with each upstream link connecting to a different service provider. Such an arrangement provides redundancy, higher network availability, and potentially better connectivity than having a single upstream link. However, current IP routing technology for this requires that the campus either have a single routing prefix that is advertised separately by each provider or that there be separate routing prefixes for each provider.[RL93] In the former case, the specific prefix (and prefix length) for that campus will need to be added to the global routing table's default-free zone, but fail-over from one provider to another (e.g. because of a fibre cut) will not impact existing network sessions. In the latter case, the prefixes can be aggregated under the respective provider's primary prefix without adding any entries to the default-free zone, but in the current Internet architecture any exiting sessions using the failed provider's routing prefix will suddenly fail.

Support for each additional routing prefix in the default-free zone increases costs and has other adverse operational impacts on all service providers around the globe, including those that do not have a given multi-homed campus as one of their customers.[RL93] For example, each routing prefix in the default-free zone needs to be carried in the routing tables of each inter-domain router on the globe. That prefix consumes memory in the routing table and forwarding table, but more importantly is suspected of having subtle adverse impacts on the path-vector routing algorithm. The existing size of the IPv4 default-free zone, more than 130 000 entries, is already cause for concern within the Internet operations community.

So a new architecture is needed that would eliminate the need to advertise any additional prefixes in the default-free zone and would also provide session continuity even when an upstream provider had a failure.

If one had a non-topological network-layer *identifier*, then that could be used instead of the IP address by the transport-layer protocol(s) and also by the upper-layer protocol(s). This would reduce the smooth-failover problem to one of notifying the remote correspondents of the still working routing

prefixes associated with the communicating host on the multi-homed campus. Various mechanisms for this can be imagined, for example new Internet Control Message Protocol (ICMP) message types could be created. Since one can use the IP Authentication Header with ICMP, such control messages could also be cryptographically authenticated.[Atk95a] Absent cryptography, the use of a nonce in a control protocol could be used to provide the same level of (in)security provided in the current IPv4 Internet.

3.3 Edge Network Issues

In the early 1990s, as the web initially took off and as major online services, for example America Online (AOL), joined the Internet, the size of the IPv4 routing table in the default-free zone grew quite rapidly. In large part, this was due to the practice of allocating address blocks directly to end-users, rather than using provider-oriented addressing. This situation was mis-reported by many in the trade press as a shortage of IPv4 addresses. To resolve this concern, changes were made to the address allocation practices. Smaller organisations were required to obtain their IPv4 address space on a lease-hold basis from one of their upstream service providers. Also, the regional Internet registries became much more systematic in requiring written justification before allocating new address blocks to service providers or to the few large organisations that were given direct allocations by the registries.

In parallel with these changes in the address allocation practices, the security of systems connected directly to the Internet became a larger concern. Many large organisations realised that only a small percentage of their computing systems needed to be connected directly to the Internet. Smaller organisations shared the security concerns and also found it cumbersome to obtain addresses and timeconsuming to carefully manage their address spaces. In the late 1990s, broadband Internet connectivity started to become widely available in many parts of the world. Many residential broadband users wanted to have Internet connectivity for web and email access, but also wanted to have a small home LAN with a printer and more than one personal computer.

Equipment vendors responded to these various circumstances by implementing a technique known as *Network Address Translation*. NAT is a technology that enables an edge router to modify the IP addresses of packets transiting that router. In the most common case, the interior address block comes from private address space [RMK+96] that is not globally unique, while the exterior addresses come from globally-routable address space. A closely related technique, *Network Address Port Translation* increases address utilisation efficiency by using port-based multiplexing and having a single exterior IP address correspond to many interior IP addresses.[TS00] Some vendors even claimed that NAPT was a security technique, though

scientific analysis does not support this claim.[Bel02]. Others proposed the use of NAT to provide load sharing.[SG98]

As we have seen previously, an architectural issue with the Internet protocol suite is the inappropriate use of the IP Address, most commonly by upper-layer protocols that actually only need a generic host identifier. The deployment of NAT and NAPT exacerbated these issues. In order to work with any TCP-based or UDP-based application, the NAT device needed to modify the IP header and also make corresponding changes to the TCP header. For the numerous applications that inappropriately embed the IP address inside the application, a NAT would need to make similar application-specific modifications to packets transiting the NAT. All of this in transit modification is a gross violation of the End-to-End Principle. [SRC84]. NAT and especially NAPT do not work flawlessly in real-world deployments. Such devices, particularly NAPT, break many protocols, thus restricting access from interior systems and greatly Balkanising the global Internet.

Some have asserted that if IPv6 deployed, NAT and NAPT will quickly disappear. Instead, it is likely that NAT or NAPT will be important transition technologies if IPv6 starts to become widely used. At present, and at least during initial IPv6 deployment, virtually all Internet content can only be accessed using IPv4. So an IPv6-only host would need to use some form of NAT device to translate the IPv6-based request into an IPv4-based request so that it could contact the content server. Further, the IPv4-based responses from the content server would need to be translated back into IPv6 so that they can be correctly understood and processed by the originating IPv6-only host.

Application-layer gateways, which are sometimes also called Proxys, are increasingly common at the edge of an enterprise network.[BBC+04] Such a gateway operates at the application-layer only, providing translation services between the interior of the enterprise network and sites outside the enterprise network. A firewall is often a specialised kind of application-layer gateway. A major issue with application-layer gateways is that the only support transmission of selected supported applications across the domain boundary. So if someone tries to use an unsupported application across that domain boundary, the attempt fails due to the application-layer gateway. In order to work properly, an proxy must create and retain session state for each session traversing the proxy. This additional state makes the network more brittle, and reduces resilience to potential failure. For example, if the routing path changes from one proxy to a different proxy, the session will fail because the second proxy does not have the session state known only to the original proxy. If security is in use, the proxy generally prevents end-to-end security. For example, if Transport Layer Security (TLS)[DA99], which is the IETF standard derivative of Secure Sockets Layer (SSL), is being used, it cannot be used end-to-end as would normally be the case. Instead, the TLS

session would need to connect the exterior system to the exterior interface of the proxy and a separate TLS session would need to connect the interior interface of the proxy to the interior host. Because of this, neither end host can be confident that the information being protected has not been modified by the proxy. Similar problems occur when many other security mechanisms, for example IP Security (IPsec), are in use. So we can see that deployment of application-layer gateways breaks the end-to-end communications model and creates security problems. If proxys and similar devices are going to be a long-term part of the deployed Internet, it might be better if they were explicitly supported by the Internet Architecture. Alternately, if they are being deployed because of a limitation in the current Internet Architecture, that would be a reason to consider evolving the Internet Architecture.

3.4 Application Issues

Because of the way that historic networking APIs have been designed, applications are required to maintain network-layer state inside the applications. Ironically, recently designed applications are perhaps the most likely to use Universal Resource Locators (URLs) containing domain names. So while the user is probably providing a DNS-based URL to the application, traditionally an application is required to perform the domain-name to IP address translation via a library call to gethostbyname(), then open the network session using the resulting IP address.

More recently, alterations to the BSD Sockets API have been proposed as part of the IPv6 development effort within the IETF.[GTBS97][GTBS99] These enhancements are intended to provide applications with a network programming interface that is independent of the Internet Protocol version in use (at least for IPv4 or IPv6). The new getaddrinfo(3) library function call takes in a host's domain name and a service name, then returns the parameters needed for the existing socket(2), bind(2), connect(2), and listen(2) system calls. These new library calls facilitate the development of applications that do not use the IP address inappropriately. However, the operating system's internal data structures, for example the Transport Control Block (TCB) contents, continue to bind the session firmly to the IP addresses. So, even though the application might be written without using the address inappropriately, the session might have difficulty handling situations where one or both endpoint addresses change during the lifetime of the session.

The common practice of using the network-layer IP address as a host identifier in various application protocols (e.g. file transfer protocol) means that many applications have protocol state that includes the IP addresses of the endpoints, rather than some more abstract and topology-independent identifier. In the case of FTP, there are two parallel communications sessions supporting a single FTP user session. One of these provides the communica-

tions channel for the control messages, while the other provides the communications channel for the file data transfer. Unfortunately, the design of FTP passes network-layer IP address information over the control channel as a host identifier. This causes traditional FTP to fail when a Network Address Translation (NAT) device exists somewhere along the path. The failure occurs because one endpoint's perception of the correct IP address is not the same as the other endpoint's perception (after NAT processing) of the correct IP address value. In this situation, users need to use passive-mode FTP to side-step this problem. A better protocol design would have passed some topology-independent identifier across the control channel, instead of a raw IP address.

This requirement for applications to know and handle network-layer objects also means that most applications would need to have additional special-purpose software to handle any situation where the IP address of the remote end might change during the communications session. This might occur when one end is a mobile host and its IP address changes as the host moves. Alternately, this might occur when a Network Address Translation (NAT) device is located somewhere along the communications path between the communicating session endpoints.

3.5 Distributed Computing Issues

Distributed computing was a major research area in the late 1980s, as the computing world tried to find more effective ways to connect and share computing resources. Distributed computing includes a broad range of capabilities. A fundamental capability is sharing files among a set of computers. In some distributed computing environments, support for remote procedure calls or process migration was included. In many distributed environments, for example MIT's Athena Project, user authentication was a key requirement and capability.

In the late 1980s, few hosts or file servers were mobile, so it was acceptable that the distributed computing protocols, including the distributed file systems, only work when the communicating hosts were at fixed locations, with fixed IP addresses. However, in the modern day, highly mobile laptops are commonly used as hosts or distributed computing clients. Also, mobile servers exist in some environments. So that compromise of the late 1980s is no longer an acceptable constraint on distributed computing systems.

Today there is strong interest in both mobile computing and ad-hoc computing. People would like to be able to use their laptops as part of a distributed computing even as they move around a campus or even move off-campus. The desire is both to support remote access to fixed file servers and also to support remote access to other mobile hosts' files. Similarly, people would like to be able to combine their own local laptop with other mobile computing resources to form an ad-hoc computing cluster. This interest is

more than academic, military forces and commercial firms (e.g. petroleum exploration firms) also are interested in these new models of computing and networking.

Existing distributed file systems, such as Sun Microsystem's Network File System (NFS) or Carnegie-Mellon University's Andrew File System (AFS), normally use domain-names as part of the user interface to the distributed file system.[CPS95] In the case of NFS, this is primarily exposed via the user configuration interface. In the case of AFS, this is exposed both via the user configuration interface and also in the filenames themselves. With AFS, the full pathname of a file typically starts with the string `/afs` which is then immediately followed by the fully-qualified domain name of the AFS cell being accessed, for example `/afs/cmf.nrl.navy.mil/`. However, implementations of NFS and AFS are limited by the same constraints as other networked applications. In order to use the network API, the implementation must translate the fully-qualified domain-name into an IP address, and then use only that IP address with the network API.

Parallel Virtual Machine (PVM) [3] is a system for building a virtual parallel computer out of heterogenous computers connected via ordinary Internet networking technology. PVM was primarily developed at the (US) Oak Ridge National Laboratory and the nearby University of Tennessee at Knoxville. PVM uses a message-passing model to build its virtual computer system. PVM has its own "middleware" protocols that are built upon standard Internet technology, IP, UDP, and TCP. In these middleware protocols, the PVM software keeps tables that include the IP address and UDP port number that are used to reach other PVM daemons. Unfortunately, in doing so the PVM application falls into the now familiar trap of using the IP address as a host identifier. So PVM does not work well in situations where one or more PVM nodes in a given virtual computer instance happen to be mobile.

Significant improvements in distributed computing could be made if one had a topologically-independent host identifier and an improved network API, so that changes in a node's IP address did not affect the continuity or security of existing or new communications sessions. Also, the existence of such an identifier could be leveraged by the distributed computing protocols themselves. For example, at present key management for distributed computing has to use an IP address or perhaps a domain-name as its host identifier. However, in many cases a single domain-name identifies a set of hosts (e.g. behind some sort of network load-balancing device) that do not share network session state among themselves. In such a case, the fully-qualified domain name is not always suitable for the distributed computing schema. Use of this identifier could be helpful in providing a namespace alternative or in distinguishing different hosts that happen to share the same

[3] More on PVM is available online at http://www.csm.ornl.gov/pvm

fully-qualified domain-name.

3.6 Security Protocol Issues

Kerberos is one of the most widely deployed distributed authentication systems today. Originally developed as part of MIT's Project Athena in the late 1980s, Kerberos continues to evolve today. Initial deployment of Kerberos was at MIT, followed by deployments at a number of other large universities, a few businesses, and at least one large Internet Service Provider. More recently, Microsoft has added Kerberos support into its Windows operating systems; also, Apple has added Kerberos support into the Macintosh OS X operating system. So Kerberos has non-trivial deployment today and is supported by most deployed computer servers and workstations. Kerberos was standardised by the IETF in the early 1990s.[KN93]

In most respects Kerberos is oriented around the combination of the user's userid and the Kerberos realm that the user belongs to. Neither of these is normally an IP address. So Kerberos is conceptually free of the misuse of the IP address as an identifier. However, there are some places where the Kerberos v5 protocol operations normally bind one or more IP addresses into a Kerberos ticket, which means that in practice Kerberos does not work well with mobile hosts if the host's IP address should change during the lifetime of a Kerberos session. Section 2.5 of the Kerberos specification notes that "Kerberos tickets are usually valid from only those network addresses specificaly included in the ticket". While Section 2.6 of the Kerberos specification discusses how IP addresses normally are bound into ticket-granting tickets "to complicate the use of stolen credentials", it also says that tickets can be issued that do not contain IP addresses. In practice, nearly all Kerberos deployments do bind IP addresses into all tickets, a decision intended to increase the effective security of the deployment. If a system whose address is not present in the ticket attempts to use a ticket containing an address, then the Kerberos server is supposed to return a KRB_AP_ERR_BADADDR message and deny the attempted use. There are numerous examples in the Kerberos specification where the address of the requesting system is compared with the address(es) in the applicable ticket as part of deciding how to respond to some Kerberos packet. So in practice, Kerberos does use the IP address as a host identifier in numerous places. Kerberos would better support mobile clients and mobile servers if it could bind to some non-topological host identifier instead of binding to the host's IP address(es).

When the IP Security (IPsec) protocols, the Encapsulating Security Payload (ESP) [Atk95b] and the Authentication Header (AH) [Atk95a] were being developed in the early 1990s, the designer faced a dilemma. On the one hand an IPsec Security Association, which is the set of key material and other configuration parameters for a secure IP session, needs to include the

identity of the originator and of the responder. On the other hand, there was not a good host identifier to use. A domain name might name a set of systems rather than a single system, while an IP address names a single network interface rather than naming a single system. In the end, IPsec can support several different kinds of identities, but none of them are ideal. The set of supported identity types are defined in the IP Security Domain of Interpretation (IPsec DoI) for the Internet Key Exchange (IKE). [Pip98] The most commonly deployed identities are iP addresses, which means that IPsec generally does not work through a Network Address Translation (NAT) device. It also means that IPsec does not work well with mobile hosts, nor can it take full advantage of the several interfaces of a multi-homed host. However, within the Internet architecture of that time, there was no truly suitable host identifier that could be used. If the Internet architecture were enhanced to have a new identifier that named a single host, not a cluster, and not a single network interface, that would enable native IPsec to support multi-homed hosts, mobile hosts, and NAT well. Additionally, the key management protocols used with IPsec, for example the Internet Key Exchange (IKE), could use such a new identity to enable them to also work well with multi-homed hosts, mobile hosts, and NAT devices.

Existing firewalls often have address-based security policies. For example, packets to a certain IP subnet (IP prefix + netmask) might need to be carried via tunnel-mode IP security. Alternately, communications to or from a particular internal IP address and port might be allowed or denied. In a new architecture, there would be some benefits. For example, the firewall could know to permit all communications from a particular mobile device (e.g. faculty laptop computer) that might happen to be on the exterior side of the firewall, regardless of which IP address that device happens to be using at the moment. However, decoupling address and identity would mean that the firewall would need to be enhanced to permit identity-based policies, instead of relying on the currently overloaded semantics of the IP address.

So we can see that security protocols and technologies are currently limited by the lack of a suitable host identifier and that a number of significant enhancements would accrue if such an identifier were added to the Internet architecture. This brings us to the question of what such an identifier might look like and what set of changes to the Internet protocol suite might be needed to enable such an identifier. This is discussed next.

4 Prior Work & Other Research

In the late 1990s, the Internet Research Task Force (IRTF) created the Namespace Research Group (NSRG) to consider whether to add a topologically-

independent identifier to the Internet Architecture.[4] This question was contentious within the group and no consensus was ultimately reached on the issue. However, as part of the group's discussions, a sub-group led by J. Noel Chiappa (and including this author) came up with the idea of a "stack name". The sub-group proposed that the Internet needed to be able to add a namespace to name each network stack. In this concept, a single host might have more than one network stack, for example in a multi-level secure (MLS) operating system. Also, for example within a distributed system, one might have a single network stack but multiple central processing units (CPUs). The full details of the new namespace were never sorted out by the sub-group – and the NSRG as a whole was unable to reach any consensus on adding a namespace. This work is a direct follow-on to that prior work.

Another spin-out from the IRTF's NSRG was a proposal from Robert Moscowitz to add a cryptographic identifier as a new namespace. This proposal, known formally as the Host Identity Protocol (HIP), is the subject of current work by the IRTF's new HIP Security (HIPSEC) Research Group and also by the IETF's HIP Working Group. In this proposal, the new non-topological identifier for a host is a cryptographic hash of the public key of the host. This is useful since one can easily authenticate that a given identity is associated with a given public-key, through knowledge of the appropriate public/private key pair. However, it simultaneously creates a significant limitation. If a private key were ever compromised, then the corresponding host would need to obtain an entirely new identity, because the identity is tightly bound to the public/private key pair used by the host.

This work proposes creating a new identifier, which is described in more detail in the next section. However, the author believes it unwise to use a host's current public key to create the host's identity. Instead, the author wants to have an identifier that can be authenticated, but that can also be preserved even if the host should change its public/private key pair.

5 A New Identifier

In this section we propose to alter the Internet Architecture to add an explicit non-topological network-layer identifier. Further, this section will describe the candidate new network-layer identifier, describe its properties (and some explicit non-properties), and provide a basic description for how the new architecture would work. The goal of this change is to enable the IP Address to resume its original function as a network-layer identifier used solely for packet routing. By doing this, we resolve many of the current application issues, the mobile networking issues, and distributed computing issues of the current Internet Architecture.

[4]The author was a member of the IRTF NSRG.

In order that any host on the global Internet can communicate dependably with any other host, it is important that any identifier used in communications sessions be globally unique. In practice, this requirement can be weakened a little. The actual requirement is that the identifier have a very high probability of being globally unique, so that any given attempted communications session has a very high probability of succeeding. In turn, this implies that the identifier needs to be sufficiently large to permit each end system to have a different identifier.

If any new identifier were created, some mechanism for assigning and distributing that identifier to end hosts would also be needed. Ideally, the selected identifier would already be available on typical computing platforms, so that no additional work would be needed to assign and manage that identifier.

5.1 Basic Properties

We propose a new identifier that is an opaque 64-bit quantity, with no heirarchy or embedded topology. It is nearly globally unique – sufficiently so that a given host is extremely unlikely to ever communicate with more than one host at a time for a given identifier. Alternately put, the intent is that this new identifier would be globally unique, but the design would not fail if that property did not strictly hold in all cases.

In short, we propose re-using the IEEE 1394 MAC Address as the new host identifier. The IEEE defined the MAC Address for use in FireWire to contain the existing 48-bit IEEE 802 MAC Address range as a proper subset. So any host that has an Ethernet (IEEE 802.3), Token-Ring (IEEE 802.5), Token-Bus (IEEE 802.4), FDDI, Resilient Packet Ring (IEEE 802.17), or FireWire (IEEE 1394) interface already has at least one identifier already assigned to it. Because a host might well have more than one such interface, it is permitted that a single host have multiple identifiers at the same time. The IEEE 1394 MAC Address is globally unique, unless a host sets the local-significance bit and creates its own non-unique identifer, a host is deliberately mis-configured with a MAC Address not assigned to it by the manufacturer, or the network interface manufacturer accidentally created more than one network interface with the same MAC Address.

It is important to remember that, in the proposed network architecture, this new identifer is a host identifier, not a link-layer interface identifier. So, for example, a host with multiple valid identifiers will normally select a single common identifier for use with all communications sessions. This means that the identifier derived from the Ethernet MAC Address might well be used on network sessions that do not communicate over that particular Ethernet interface.

6 Systems Architecture

The previous section described the proposed new network-layer identifier in detail. This section describes how the current Internet Architecture and some of the key existing protocols would need to evolve both to support this new identifier and also to resolve the issues identified in the first section of this paper.

6.1 DNS Enhancements

A new resource record, the ID record, is added to the Domain Name System within the Internet (IN) zone. This record is associated with a fully-qualified domain name, such that given a fully-qualified domain name one can use DNS to find the set of associated identifiers. Each ID record contains a single 64-bit network-layer identifier. A given fully qualified domain-name might have more than one valid ID record at a time. Other IETF work in progress provides a mechanism to place a host's public key into the DNS.

If one is going to deploy any identifier it is important to be able to authenticate the bindings between that identifier and other parameters associated with the legitimate holder of that identifier. Fortunately, recent work on authenticating the Domain Name System makes this straight-forward. [Wel00b, Wel00a, 3rd01] DNS Security extensions are used to cryptographically bind the ID record to the domain-name, and also to bind the host's public key to the domain-name.

6.2 IP Enhancements

In the currently deployed Internet, most sessions do not use any form of cryptographic security. So those sessions are not protected against an active or passive attack by some unauthorised party along the path between the communicating parties. To reduce the operational risk in the new architecture to that same level, a new IP option is added. This new end-to-end option contains a *session nonce*.

The session originator includes the nonce in the first packet of the session. The responder includes the nonce in its first reply packet. From that point onwards, the session nonce is used in all control messages for risk reduction. This is implemented as an IP-layer option, but it could instead be implemented as a transport-layer option or as a control protocol message. The IP-layer option appears to be the best implementation choice. If implemented in the transport-layer, the same capability would need to be added to each transport protocol, which would mean the same capability would be present on the wire in several forms and which would also mean that changes to the networking protocol suite would be more widespread. If implemented as a new control protocol message, there would be a significant

increase in the deployment difficulty, because many sites use firewalls that block all control protocol messages or block all unknown (e.g. new) control protocol messages.

6.3 ICMP Enhancements

The Internet Control Message Protocol (ICMP) is enhanced with two new message types in this proposal. Each of these messages includes the *session nonce* established at the beginning of the communications session to reduce the risk of forgery. Additionally, standard techniques such as the IP Authentication Header (AH) can be used to provide cryptographic authentication to this or any other ICMP message.

The first message is the ICMP *I Have Moved* message, which is used to provide a hint to an existing correspondent that the message originator has a new IP address. Using this message helps the remote correspondent remain aware of the current IP address to use even as the local correspondent's IP address might change (e.g. due to change in network location). This is primarily intended for use by mobile hosts.

The second new ICMP message is the ICMP *Preferred Routing Prefix* message, which is used to suggest a different routing prefix that the correspondent should use in communicating with the host originating that ICMP message. This is primarily intended for use by multi-homed hosts, either directly multi-homed with multiple network interfaces or indirectly multi-homed via a local campus that has more than one Internet uplink.

6.4 Other Protocol Modifications

The transport-layer protocol implementations would be modified to use this new 64-bit host identifier value in each place (e.g. Transport Control Block) where the IP Address is used as an identifier at present. Also, the current Session Nonce would likely be cached in those same places in the implementation. Equivalent modifications would be made to each upper-layer protocol and application. So if this concept were fully implemented, the IP Address would cease to be used by any upper-layer protocol or ordinary application. So the IP Address would return to its original role in routing packets. The operating system software that implements the network-layer and transport-layer would need to be modified to support the new ICMP messages noted above.

For a given transport-layer session, the identifiers in use for that session must not change during the lifetime of that session. Normally, a given host will always use the same identifier for all communications sessions. However, it is permitted that if a given host (A) has multiple valid identifier values, then A might use different network-layer identifiers to represent A in different communications sessions with another host (B) at a given point in

time. In order that the binding between domain-name and identifier can be authenticated, along with the bidirectional binding of domain-name and IP Address(es), it is important that the relevant data all be stored correctly in the DNS and also that DNS Security be enabled. Because the identifier is not related to the IP address, the host's IP address(es) may change during the session without adversely impacting the session.

This new systems architecture requires that the Domain Name System be operating properly in order for applications to initiate new communications sessions. A properly operating DNS is not a formal requirement of the current Internet, though most users of the current Internet are unable to distinguish between a DNS failure and a general network failure. If an application that follows the new systems architecture is in use and the DNS fails, the application probably will not be able to initiate new communications sessions.

7 Open Issues & Future Research

Although the proposed architectural and protocol modifications do usefully mitigate the issues identified at the beginning, there are some side-effects that one needs to be aware of. Also, there are some unresolved issues, such as support for multicasting, that are good candidates for future research. This section will elaborate on both of these topics. Finally, a more detailed comparison of this proposal against other proposals, such as HIP, needs to be performed.

7.1 Increased Reliance on DNS

As the Domain Name System was not deployed until several years after the Internet protocol was deployed, many aspects of the Internet can still work when the DNS is not working or is not working well. Prior to the deployment of the web, most users were relatively sophisticated and could recognise the difference between a network-layer failure and a DNS failure, so they could often work around a DNS fault. However, in recent years most users are primarily interested in accessing network-based content, such as the web. For most such users, a DNS failure is equivalent to the Internet being down. [5] So during the past decade, the majority of the user base has become quite dependent on the DNS.

This proposal has the side effect of making all users much more dependent on the DNS working properly. If one misuses an IP Address as a host identifier as at present, then a DNS failure will not necessarily impact the ability to use the network – provided one already knew the remote end's

[5] In fact, DNS failures are often reported to consumer ISPs as "network down", much to the frustration of the ISP support staff.

IP address. However, if one moves to an architecture and to revised protocols that use the proposed new host-identifier and/or fully-qualified domain names as the principal identifiers above the network-layer, then it will be much more difficult to work around a DNS fault. In metallurgy, when one makes a metal that is stronger, the result is typically both stronger and more brittle. So, to draw an analogy, the proposed new architecture and revised protocols strengthen the Internet and perhaps also make it more brittle.

It is, however, important to note that many critical network debugging tools, such as ping(1), traceroute(1), or nslookup(1), should continue to work well without modification. This does matter operationally, as any distributed system will suffer faults from time to time; it is important that when a fault occurs, suitable tools exist for analysing the cause of the fault, for repairing the fault, and for mitigating the fault's effects until repaired.

7.2 Multicasting

In the special case of a multicast session, the current proposal does not describe what identifier should be used to represent the multicast group. While some transport protocols, such as TCP or SCTP, are not used with multicasting, other protocols, primarily UDP and RTP, are regularly used with multicasting.

This is a significant short-coming of the current proposal; this proposal cannot be considered complete unless multicasting is supported in some manner.

7.3 Security

While this proposal has not ignored security considerations in its design, there is no well-documented threat analysis for the new architecture – nor is there a threat analysis for the requisite protocol modifications. This proposal cannot be considered complete without such a threat analysis, describing potential threats, security mechanisms used to mitigate those threats, and the residual risks of the new architecture. While the author believes the net effect of the new architecture will be to facilitate wider deployment of better quality security mechanisms than are deployed today, this should be documented and shown to actually be the case.

7.4 Prototyping & Experimentation

The experience of the IRTF NameSpace Research Group (NSRG) indicates that while some will be able to understand the new architecture and system after reading documentation, many others will not believe that the system could actually work absent experimental results. So once the design is stable and reasonably complete, it would be sensible to build a prototype imple-

mentation, deploy it at least in a small testbed, and run suitable experiments to validate the architecture and also the protocol specifications.

If this issue is addressed via implementation, the work would likely be most accessible to others in the networking research community if implemented inside a networking software stack derived from 4.4 BSD Unix. If this is addressed using a network simulation instead, perhaps because of lack of resources/time to do a full implementation, then the work would be probably most accessible to others if coded using the NS network simulator or using the commercial OPnet simulation tool.

8 Summary

This working paper has outlined a set of technical issues and limitations with the current Internet architecture. The issues described include several specific difficulties in the deployed Internet of today. Further, this paper proposes that the Internet architecture would be improved by adding a new host identifier that is not used by the network routing system (i.e. is not an IP address) to mitigate or resolve the identified issues. We have also outlined other changes needed to existing protocols to fully support the enhanced network architecture. Finally, we described the open issues and several areas that appear worthy of future research.

References

[3rd01] D. Eastlake 3rd. RSA/SHA-1 SIGs and RSA KEYs in the domain name system (DNS). RFC 3110, Internet Engineering Task Force, May 2001.

[Atk95a] R. Atkinson. IP authentication header. RFC 1826, Internet Engineering Task Force, August 1995.

[Atk95b] R. Atkinson. IP encapsulating security payload (ESP). RFC 1827, Internet Engineering Task Force, August 1995.

[BBC+04] A. Barbir, E. Burger, R. Chen, S. McHenry, H. Orman, and R. Penno. Open Pluggable Edge Services (OPES) Use Cases and Deployment Scenarios. RFC 3752, Internet Engineering Task Force, April 2004.

[Bel02] Steven M. Bellovin. A technique for counting natted hosts. In *Proceedings of 2nd Internet Measurement Workshop*. ACM SICOMM & USENIX, November 2002.

[CPS95] B. Callaghan, B. Pawlowski, and P. Staubach. NFS version 3 protocol specification. RFC 1813, Internet Engineering Task Force, June 1995.

[DA99] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, Internet Engineering Task Force, January 1999.

[GTBS97] R. Gilligan, S. Thomson, J. Bound, and W. Richard Stevens. Basic socket interface extensions for IPv6. RFC 2133, Internet Engineering Task Force, April 1997.

[GTBS99] R. Gilligan, S. Thomson, J. Bound, and W. Richard Stevens. Basic socket interface extensions for IPv6. RFC 2553, Internet Engineering Task Force, March 1999.

[HHM71] E. Harslem, J. Heafner, and E. Meyer. Request for comments on socket name structure. RFC 129, Internet Engineering Task Force, April 1971.

[KN93] J. Kohl and C. Neuman. The kerberos network authentication service (V5). RFC 1510, Internet Engineering Task Force, September 1993.

[LMKQ89] Samuel J. Leffler, Marshall Kirk McKusick, Michael J. Karels, and John S. Quarterman. *"The Design and Implementation of the 4.3 BSD UNIX Operating System"*. Addison-Wesley, Menlo Park, CA, 1989.

[MBKQ96] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman. *"The Design and Implementation of the 4.4 BSD UNIX Operating System"*. Addison-Wesley, Menlo Park, CA, 1996.

[Moc83] P. V. Mockapetris. Domain names: Concepts and facilities. RFC 882, Internet Engineering Task Force, November 1983.

[Moc87] P. V. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, November 1987.

[NKPC70] J. Newkirk, M. F. Kraley, J. B. Postel, and S. D. Crocker. Prototypical implementation of the NCP. RFC 55, Internet Engineering Task Force, June 1970.

[OY02] L. Ong and J. Yoakum. An introduction to the stream control transmission protocol (SCTP). RFC 3286, Internet Engineering Task Force, May 2002.

[Per02] Charles E. Perkins. IP mobility support for IPv4. RFC 3344, Internet Engineering Task Force, August 2002.

[Pip98] D. Piper. The Internet IP security domain of interpretation for ISAKMP. RFC 2407, Internet Engineering Task Force, November 1998.

[Pos81a] J. B. Postel. NCP/TCP transition plan. RFC 801, Internet Engineering Task Force, November 1981.

[Pos81b] J. B. Postel. Transmission control protocol. RFC 793, Internet Engineering Task Force, September 1981.

[Pos81c] John Postel. Internet control message protocol. RFC 792, Internet Engineering Task Force, September 1981.

[RL93] Y. Rekhter and T. Li. An architecture for IP address allocation with CIDR. RFC 1518, Internet Engineering Task Force, September 1993.

[RMK+96] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, February 1996.

[SG98] P. Srisuresh and D. Gan. Load sharing using IP network address translation (LSNAT). RFC 2391, Internet Engineering Task Force, August 1998.

[SRC84] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[TS00] G. Tsirtsis and P. Srisuresh. Network address translation - protocol translation (NAT-PT). RFC 2766, Internet Engineering Task Force, February 2000.

[Wat71] R. W. Watson. NIC view of standard host names. RFC 237, Internet Engineering Task Force, October 1971.

[Wel00a] B. Wellington. Domain name system security (DNSSEC) signing authority. RFC 3008, Internet Engineering Task Force, November 2000.

[Wel00b] B. Wellington. Secure domain name system (DNS) dynamic update. RFC 3007, Internet Engineering Task Force, November 2000.