TRIAD:
A Scalable Deployable NAT-based Internet Architecture

David R. Cheriton, Mark Gritter
Computer Science Department
Stanford University
{cheriton,mgritter}@dsg.stanford.edu

Abstract

Network address translation (NAT) has become an important technology in the Internet, supporting scalable addressing, addressing autonomy, concealed endpoint identity, and transparent redirection. However, NAT currently lacks a well-specified scalable architecture and interferes with end-to-end security and reliability.

In this paper, we present TRIAD as a NAT-based architecture that solves these problems. The key ideas of TRIAD are: i) basing all identification on DNS names, not end-to-end addresses, supported by a router-integrated directory service, ii) providing end-to-end semantics with a name-based transport-level pseudo-header, and, iii) using a simple "shim" protocol on top of IPv4 to extend addressing across IPv4 realms, localizing this extension to inter-realm gateways. We claim that TRIAD solves the problems with NAT, is incrementally deployable, and eliminates the need to make the painful transition to IPv6.

1 Introduction

The universally unique 32-bit address was a central tenet of the original Internet architecture. However, the success of the Internet has proven this approach to be inadequate, at least with 32-bit addresses.

In 1992, with the Internet apparently running seriously low on unallocated addresses, Network Address Translation [9, 10] (NAT) was introduced to effectively allow addresses to be reused. Since then, the use of NAT has proliferated along with the deployment of firewalls, not just for address reuse but for several other purposes. For one, NAT is used for *address allocation autonomy*, allowing an enterprise to assign addresses independent of the ISP, supporting multi-homing, switching ISPs and decoupling the number of hosts from the number of addresses provided by the ISP. NAT is also used to conceal endpoints when an leaf network does not want its host visible to the rest of the network. Finally, NAT is used to allow a transparent redirect, allowing a cluster of web servers to appear as a single server, a key approach to building scalable web sites. Redirect is also used to support wiretapping of VoIP-based telephone service, as required by law. In these uses, NAT is similar to the use of virtual memory address translation with operating systems, providing protection, concealment, address reuse and remapping.

With all these uses, NAT has become a valuable part of the Internet but violates the end-to-end semantics of the original Internet architecture. In particular, with NAT, an IP address is only meaningful within one *address realm*[1]. Consequently, application-specific proxies are required in NAT routers to make some Internet applications function correctly. For example, a NAT box must rewrite the ASCII-encoded address passed in an FTP control session as it passes across a NAT boundary. Similarly, it must modify DNS responses that transit through it. Since transport-level checksums cover the IP source and destination addresses, a NAT box must also update the transport checksum of a packet even if it is only changing the address of the packet, compromising end-to-end reliability and conflicting with end-to-end security. With NAT, it is also hard to communicate freely between separate private realms without renumbering, such as is needed when two NAT-based enterprise networks are merged.

As a consequence of the use of NAT, there is no clear model of how Internet applications are to be structured. If an application depends on DNS, it can fail because no DNS server is available or reachable

[1] The term *realm* [15] is used as it is used with current NAT, to designate a collection of interconnected hosts across which the IPv4 addresses are unique.

even if the communication paths to the application endpoints are functioning. If it uses IP addresses directly, it can fail because of the changing address assignments and translation occurring between endpoints that can arise with NAT, unless an application-specific proxy is deployed at all NAT points. Yet, this proxy precludes end-to-end encryption of the packets and exposes the application to undetected corruption because of a failed or compromised proxy. These *ad hoc* mechanisms will become even harder to manage as network configurations are forced to use multiple levels of NAT to further scale addressing and there are more applications and more NAT routers.

In this paper, we describe TRIAD[2], a NAT- and IPv4-based Internet architecture that addresses the problems of NAT. TRIAD provides end-to-end reliability and security by using a name-based transport-layer pseudo-header. TRIAD extends the addressing available with conventional NAT using a shim protocol over IPv4 called WRAP. And TRIAD provides an router-integrated directory service that supports multicast channels, mobility, virtual private networks and policy-based routing. TRIAD can be incrementally deployed without any changes to end-hosts or applications beyond that already required for NAT. The scalability of TRIAD eliminates the need to transition the Internet to IPv6.

2 TRIAD Overview

In TRIAD, as with conventional NAT, the Internet consists of an interconnected set of address realms connected in a hierarchy. At the leaf level, an address realm corresponds to an enterprise or university network, a military installation, or much smaller units like a collection of autonomous sensors or a home network or even a set of virtual hosts on a single physical machine. At this level, the firewall or border router is extended to act as a TRIAD *relay agent* between realms, translating packet addresses as it relays packets between the realms that it interconnects. Higher-level address realms correspond to local and global Internet service providers (ISPs). Backbone or wide-area ISPs can connect at peering points, the same as today, but with high-speed relay agent routers at these points.

The end-to-end Internet-wide identification of a host interface[3] or multicast channel is provided in TRIAD by a hierarchical character-string (DNS) name. This *name* is the basis for all end-to-end identification, authentication, and reference passing. There is no other identifier for the host interface that is global and persistent, unlike addresses in IPv6 and in the original Internet architecture. In particular, (IPv4) addresses have no end-to-end significance.

Within a realm, the operation of naming, addressing and routing operates the same as currently with IPv4. Thus, there are no host or router changes required. The relay agent or agents connecting a realm to other realms provide naming and routing services plus WRAP relaying of packets. WRAP, the *Wide-area Relay Addressing Protocol (WRAP)*, is a "shim" protocol which carries the transport header and data as its payload, similar to other IP encapsulation protocols. The WRAP header contains a pair of *Internet Relay Tokens (IRTs)*, the *reverse token* and *forward token*. An IRT is a potentially opaque variable-length field that extends the addressing beyond that provided by IPv4.

Fig. 1 illustrates the operation of TRIAD between realms with two hosts, `src.Harvard.EDU` and `dst.Ietf.ORG`, assuming `Harvard.EDU` and `Ietf.ORG` are two separate realms connected via a single intermediate realm, the "external" Internet. For `src` to send to `dst`, the name lookup of `dst.Ietf.ORG` is handled by the relay agent `relay.Harvard.EDU` for this realm, with internal IPv4 address $RA1$ and external IPv4 address $RA1'$. This relay agent determines the appropriate next relay agent from its directory mapping of `Ietf.ORG` and then communicates the name lookup across the Internet to the `relay.Ietf.ORG`, the relay agent for the `Ietf.ORG` realm. (This relay agent has internal IPv4 address $RA2$ and external IPv4 address $RA2'$.) In response to this query, `relay.Ietf.ORG` returns to `relay.Harvard.EDU` an IRT `f'` that designates `dst` relative to $RA2'$. Then, `relay.Harvard.EDU` returns an IRT `f` to `src` which designates `dst.Ietf.Org` relative to $RA1$, creating any state it needs to map `f` to `f'`.

Then, `src` sends an IPv4 packet addressed to $RA1$ with `f` stored in the WRAP header. On reception, $RA1$ translates `f` into `f'` and transmits the packet with destination address $RA2'$ and source address $RA1'$, as shown in the middle packet in the figure. The WRAP header also contains the reverse IRT `r'`, which indicates the source of the packet relative to $RA1'$.

[2]TRIAD is an acronym, originally standing for *Translating Relaying Internet Architecture integrating Active Directories* but it might also stand for *Time to Rescue the Internet from Address Depletion.*

[3]Packets are addressed to host interfaces, not to hosts, the same as the original Internet architecture.

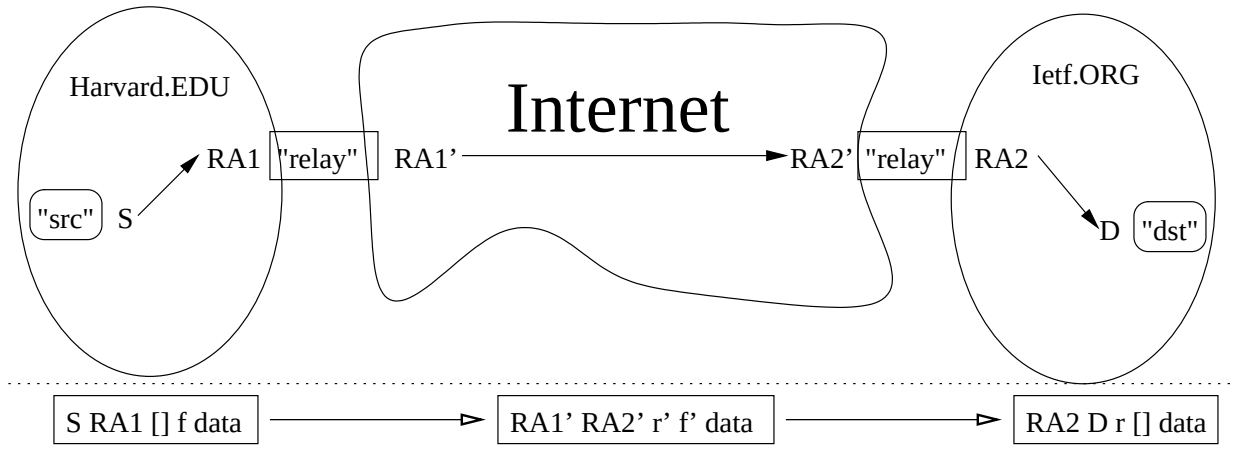

Figure 1: Inter-realm packet transmission in TRIAD: The host named "src" with IPv4 address S in realm Harvard.EDU sends to the host named "dst" and IPv4 address D in realm Ietf.ORG. The packets below the dotted line indicate how the IP and WRAP headers changes as it crosses the three realms, with the header listed as source address, destination address, reverse IRT, and forward IRT.

On reception at $RA2'$, the reverse IRT in the packet is translated to a new value `r` which represents `src.Harvard.EDU` relative to $RA2$. This relay agent then transmits the packet with an empty forward IRT, IPv4 destination address D and and source address $RA2$, as shown in the rightmost packet in the figure.

Thus, `dst` receives the packet as a normal IPv4 packet sent by its relay agent $RA2$ but also containing an IRT that identifies the actual source of the packet relative to its relay agent. The packet is then passed up to the next higher layer processing, such as TCP or UDP.

The destination can determine the source host name by a reverse name lookup at the last relay agent using the received IRT. However, it can respond to a packet directly by "reversing" the IRT (using a fixed algorithm) and sending the packet to the local RA with this reversed IRT. This causes the packet to return along the reverse of the relay path on which the original packet was received. Alternatively, the destination can perform a lookup on the source name (after determining it as above) to get a separate IRT and RA to reach this host. This alternative is more flexible, allowing for asymmetric routing at the cost of an extra name lookup.

The IRT and the RA address are local in scope and transient. That is, the IRT is only meaningful relative to the RA and is only guaranteed to be *T-stable*: it does not go from one valid association with a relay agent to another in less than time T, where T is typically hours. In particular, it can become invalid at any time but can only be reassigned to another use after time T. Thus, passing an IP address or an IRT in the data portion of a packet to the other endpoint is meaningless in general.

A WRAP proxy, referred to as a WRAPID gateway (see Section 6), allows existing IPv4 hosts to interact with WRAP-enabled hosts and servers without any modification. A WRAPID gateway is just an extended NAT-capable router or firewall which is able to WRAP and unWRAP packets going through it, as appropriate.

TRIAD provides end-to-end transport semantics while still making extensive use of address translation by basing the transport-layer pseudo-header on names, not addresses, as described next.

3 End-to-end Semantics

In TRIAD, the transport layer checksum covers the name of the source and the name of the destination and *does not* include the addressing in the packet. In the case of multicast, the pseudo-header is based on the name of the channel. The addressing allows efficient forwarding of the packet to a destination; the name-based pseudo-header ensures it arrived at the correct destination (even though the names are not present in the header). Thus, a transport connection is between two endpoints identified by name, not address. To support these semantics, TRIAD-TCP operates as described below.

TRIAD-TCP takes a name parameter as part of connection setup to designate the destination, rather than an address. On connection setup, the local endpoint computes and saves a *precomputed pseudo-header checksum value (PHCV)* based on the name of the source and the destination, similarly at the remote endpoint. The source and destination names are also stored in the connection state record with the PHCV. When a packet is transmitted, the checksum is computed starting with the PHCV, effectively incorporating this name-based pseudo-header into the packet checksum. On reception, the packet is demultiplexed to the TCP connection state based on the packet addresses and port numbers. This receiving connection state contains the same PHCV, allowing the receiver to (re)compute the packet checksum efficiently. If the computed checksum fails to match that in the packet, the packet is considered corrupted in transmission.

If the packet does not map to a valid connection state, the receiver does a reverse name lookup to determine the source name and looks up the connection by name, creating it if this is a connection setup packet, e.g. a SYN packet in the case of TCP. If the name lookup fails, the packet is discarded as a corrupted packet. An endpoint may receive a packet for an existing connection that does not match based on addressing if the destination name has been rebound to a new IRT (perhaps because of a rerouting or relay agent reboot). The name-based mapping allows the connection state to be located and the address mapping to be corrected.

A transport-level checksum based on this pseudo-header provides end-to-end reliability because it detects corruption of the packet addresses in transit yet does not need to be modified in transit as part of relaying (or forwarding) the packet even though the packet addresses are modified.

This name-based pseudo-header checksum also allows end-to-end security because, with the relay agents not modifying the packet data or checksum, there is no conflict between an encrypted packet and network address translation. WRAP includes a security mechanism similar to IPsec, WRAPsec, which uses names for identification and the same pseudo-header for integrity check verification (ICV), and provides end-to-end security. Note that dispatch to connection state before validating the checksum is required both for secure and insecure connections, unifying the processing in both cases. With conventional TCP implementations, the checksum is often checked before mapping to the associated TCP connection state.

The endpoint name, stored with the connection state, is used to re-establish the connection if the connection fails. This re-establishment works because the name stored by the connection is the proper name for the other end. That is, the rebinding, if successful, maps to the endpoint that is storing the transport-level state of the connection, allowing the transport-level connection to continue with the new address binding, similar to an ARP-level rebinding. The only issue is identifying the state, and that relies on the name, which does not change.

Using this mechanism, TCP can transparently recover from changes in the relay path, whether caused by relay agent reboots or link failures (assuming an alternate path is available). This rebinding makes the translation in the network effectively *soft state*, preserving one of the key properties of the Internet. UDP-based services are expected to similarly rebind from the name, either periodically or on timeout, in the case of a reliable protocol built on UDP.

Hosts such as public web servers may want to respond to a connection request without (synchronously) looking up the client's name. To support this behavior, TRIAD-TCP includes a (new) option that can carry the PHCV, allowing the server to use this value as the base for the packet checksum calculation. Otherwise, a host receiving a connection setup request does a reverse name lookup on the RA and IRT in the packet to determine the source name. For backward compatibility, TRIAD-TCP uses the current TCP checksum algorithm for a connection where the source and destination are in the same realm, i.e. the IRT is null. However, this option can also be used to negotiate the use of the name-based pseudo-header checksum in general even without use of WRAP.

TRIAD-TCP provides an negotiated "unreliable" mode which simply disables retransmissions. This minor extension to TCP as a negotiated option allows applications such as real-time VoIP and video to use TCP and automatically get the rebinding behavior described above (as well as the TCP congestion avoidance mechanisms). With this provision, TCP can replace the wide-area use of UDP in all cases that we are aware of. Then, UDP is only used local to a realm, if at all.

TRIAD routers include support for WRAMP[4], an ICMP-like protocol for sending "destination unreachable" messages, similar to ICMP, thus informing hosts (on a best efforts basis) when an (RA,IRT) pair is

[4]The *Wide-area Relay Addressing Management Protocol*

no longer valid. The behavior in TCP of allowing infinite timeouts when the connection is idle is supported in TRIAD by a timestamp on the name stored in the connection record, and rebinding when a connection becomes non-idle if the connection has been idle with no relookup for some excessive period of time.

In TRIAD, name mapping is made as reliable as packet forwarding, so applications can use names without loss of reliability. Also, name mapping is as secure as packet forwarding so there is no more risk of the name mapping providing the wrong host than relying on its IP address. This secure, reliable integrated directory service is needed so applications and systems can trust both forward and reverse name lookup. Of course, higher-level checksumming, encryption, and authentication can provide additional security and reliability when needed.

With these techniques, applications on WRAP-aware hosts using TCP have end-to-end semantics and are oblivious to loss of relaying and translating state because of relay agent reboot, except possibly for the performance impact. These changes to TCP do not change the packet format, are local to the implementation, and allow unmodified hosts to communicate within a realm.

Let's now consider WRAP in more detail.

4 WRAP

WRAP [1] is a shim protocol that uses a variable-length header between the IP layer and the transport layer, providing NAT with extensible addressing.

A WRAP packet is formatted as in Fig. 2 as the payload of an IPv4 packet.

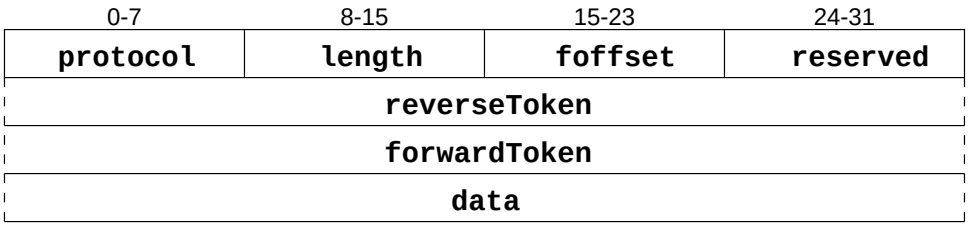

Figure 2: WRAP Packet Format

The protocol field specifies the higher-layer protocol in the "data" field using the same types as for IP, e.g 6 for TCP. The length field is the number of 32-bit *components* in the header. Thus, the WRAP header length in octets is 4 + length * 4. The foffset field is an offset into the list of components where the forwardToken starts, with 0 referring to a null reverseToken, so the forward token starts in the first 32-bit field. The forwardToken is a value used by the next relay agent to determine how to relay the packet to its destination. The reverseToken is a value used by the previous relay agent to refer to where the packet came from.

The data field contains a TCP, UDP, or other transport protocol packet.

A WRAP source sends packets to a destination by forming an IPv4 packet with the IP destination address set to the relay agent address and an appropriate WRAP header. The WRAP forwardToken contains the IRT of the destination relative to this RA, and the reverseToken is null. Thus, the length field is the length of the forwardToken and the foffset field is 0. (The foffset value is also the length of the reverseToken.)

The RA, on receiving an WRAP packet:

1. maps the (SA,DA) of the packet to a *virtual interface* (VI) that represents the local endpoint of the realm "channel" on which the packet arrived. It maps the next k 32-bit components of forwardToken field (assuming foffset is less than length) to a corresponding relay entry in a relay table associated with this VI.

2. determines from this entry the next IP source address (the "egress" interface), the next relay agent's IP address, the new forward token, and the rewrite of the reverse token to perform.

3. forwards the modified packet to the next RA, with the IPv4 destination address as that of this next RA and the IP source address determined above, and increments the foffset field.

The relay agent thus "consumes" one or more 32-bit components from the forwardToken and adds an equal number of components to the reverseToken. (However, both these fields may be translated according to the information in the relay agent's lookup table.) The reverse token, when component-reversed, must be recognized by this relay agent when used as a forward token, to send packets back toward the source.

Normally, an RA just rewrites the first forward token component and increments the foffset by 1. In the simplest case, the rewritten component is just encodes the IP source address of the incoming packet (that is, the last relay point.) This restricted form of relaying, though less general than WRAP allows, is more amenable to hardware implementation, since less of the packet needs to be rewritten.

If the RA does not recognize the forward token, it drops the packet and may send a WRAMP message back to the previous RA. The relaying state may include filters on sources from which to accept packets and destinations allowed for given sources.

The receiver of a WRAP packet is a node that receives an WRAP packet with a null forwardToken, or receives a packet with a multicast destination and subscribes to that multicast source.

The actual source of the packet is identified by the reverseToken and the last hop RA for the packet. The receiver can contact this last hop RA to do a reverse name lookup on this IRT to determine the name of the actual source, as described in Section 2.

Between relay agents, a packet is routed by the normal IPv4 routing protocols used within the realm, so WRAP is similar to loose source routing with the relay agents as the designated nodes on the path it is to follow. A packet that does not travel outside of a single address realm can (and should) omit the WRAP header entirely.

With WRAP, a packet is reassembled from fragments at each intermediate relay agent, because each relay agent is a destination from the IP standpoint. This feature reduces the risk of carrying packet fragments all the way to the destination only to discover some fragment is missing.

4.1 Transparent and Opaque Relaying

WRAP allows the relaying to be *transparent* in the sense that each IRT is simply a sequence of IPv4 addresses designating relay agents and endpoints, an *Internet Relay Path (IRP)*. The IRT can also be *opaque* so that a holder of the IRT cannot determine the relay path nor can it forge a valid IRT.

Using a transparent IRT, the RA is stateless in the sense that the relaying only relies on routing/directory state and configuration state; it does not require state to be created on name lookups. In this mode of operation, the relay node is statically configured with an IPv4 address for each of the other realms it connects to, so that an address uniquely identifies which direction to relay a packet (and a particular "egress" address in that realm.) Upon receipt of a WRAP packet, the relay agent replaces the IP destination with the first forward token component, uses the egress address as the new IP source, and places the old IP source as the last component in the reverse token. This is the simplest possible relaying action, requiring only 4 words in the packet headers to be modified.

To make the WRAP addressing opaque to an observer, the relay agent can choose to put a random value in the IRT and translate it to/from IPv4 addresses using the relay table described earlier. This opaque form prevents a upstream source from fabricating IRTs, forcing it to rely instead on the directory service to supply IRTs. In particular, an ISP can retain control of routing, preventing customers from using unauthorized routes. It also prevents a third-party observer from determining protected information from addresses in the packets.

An IRT must have the *reversibility property*, namely that the component-wise reversal of the received IRT provides an IRT that can be used to send a packet back to the source of the packet using the relay agent from which the original packet was received.

An IRT normally also has the *concatenation property*, i.e. if the IRT to a relay agent is X from a host H and Y is the IRT to a destination D relative to this relay agent, then XY is an IRT to destination D from host H. The directory indicates whether the returned IRT supports concatenation or not.

4.2 Multicast

WRAP supports the EXPRESS [3] single-source model of multicast. Multi-source multicast applications can be supported with EXPRESS by relaying the multicast through a node that is a source of an existing relay multicast channel, similar to the rendezvous point in PIM-SM, but performed at the WRAP or the application layer.

A subscriber joins a multicast channel by specifying the associated RA, reverse token, and destination multicast address. It receives this information by looking up the multicast channel by name in the TRIAD directory service.

A multicast WRAP header contains the same multicast address G repeated r times, where r represents the maximum number of relay hops in the multicast tree; this allows multicast WRAP relaying to performed identically to unicast WRAP relaying. As multicast packets are relayed, group addresses may be translated so that the (S,G) pair upon which IPv4 routers do multicast delivery is unique. However, a single intra-realm channel can be reused within a realm to deliver multiple inter-realm channels.

4.3 Secure Communication

TRIAD includes a secure communication facility similar to IPsec, i.e. end-to-end at the (inter)network layer. It differs primarily in working in the presence of NAT or WRAP translation, because the Integrity Check Value (ICV) does not include the packet addressing information, similar to the TRIAD-TCP pseudo header.

4.4 Virtual Private Networks (VPN)

Using WRAP, an ISP can provide a *Virtual Private Network (VPN)* service by creating for each enterprise a secure (virtual) transit realm that connects to each of its enterprise sites. The enterprise directory and routing system only needs to deal with the topology of the enterprise network with this "virtual" realm directly interconnecting all the sites. The different sites of the VPN can even have overlapping IPv4 address assignments (typically 10.X.X.X) yet still communicate directly without renumbering.

The ISP implements this virtual realm simply by providing routing and secure communication between each site. In this case, the overhead from WRAP is 12 bytes. As an optimization, the ISP can provide at each site a relay agent address for each other remote site in the VPN so packets can be addressed as though each site was directly connected through a relay to each other site, reducing the header overhead to 8 bytes. Thus, WRAP can also obviate the need to deploy MPLS [12].

4.5 Policy-based Routing

WRAP supports policy-based routing across multiple realms[5] by allowing the IRT to direct packets through certain relay agents and to avoid others, with the directory mapping particular names to these policies. For instance, a special name in the name lookup can provide a special IRT to a destination that directs packets over a more secure ISP network to a particular destination rather than using a cheaper but less secure route. The ISP directory service can also provide different IRTs based on the class of service that the requesting customer is paying for. WRAP relaying is similar in this sense to source routing and tunneling, but with the key differences discussed in Section 9.

This routing control can also be used for traffic engineering.

4.6 Extended Forwarding Path Check

WRAP supports an *extended forwarding path (EFP)* check based on the WRAP header indicating the (relay) path it took to the receiver, not just the port that the packet arrives on. The receiver can easily reject packets from untrusted relay agents. This check is not tied to the reverse path logic because a receiver or relay can check whether the relays that the packet took are trusted and accepted, independent of whether it would forward a packet to the source of this packet back along the same path. Unlike conventional source routing, WRAP operates with strict reverse path forwarding (RPF) checking in place and does not allow source spoofing attacks. Even with encryption techniques providing authentication, RPF checks at the IP and WRAP levels are important to prevent denial of service attacks and various forms of network device failures and misconfiguration.

Verification of packet sources and prevention of man-in-the-middle attacks are a growing concern with the scaling of the Internet. So-called source spoofing is the basis for many denial of service and security attacks. While end-to-end encryption techniques, properly deployed, can protect confidentiality and integrity and ensure authentication, they can, by their cost in processing, actually make denial-of-service a bigger problem. That is, the increased time to decrypt a packet with secure communication, only to discover it is a bogus packet, means a node loses more resources in an encrypted denial-of-service attack than with plaintext messages. (Providing wire-speed hardware-supported encryption addresses this problem in part, but is an

[5] Each realm can support its own local policy-based routing.

expensive solution for low-end systems and generally does not deal with setup processing, such PKE-based authentication on connection setup.) For mission-critical applications, denial-of-service may be as damaging an attack as any of the other possible security attacks.

4.7 Implementation and Performance

In comparison to conventional forwarding, relaying requires an additional lookup of the next forwarding component in the context of the virtual interface to which the packet is mapped with the (SA,DA) lookup. A hardware implementation may add an additional lookup resource to handle this or simply perform two lookups on the same memory, depending on the speed of this memory and the forwarding performance requirements. We expect initial hardware implementations may restrict the *foffset* they will support, throwing packets with larger values to software.

Multicast relaying uses additional state in the forwarding path but is the same implementation and performance.

Our Linux implementation of WRAP added about 1,500 lines of code as a kernel module and incurred an extra 2.2 microsecond overhead (or 2.6 percent) for relaying compared to conventional IP forwarding.[6] Thus, the complexity is minimal for either software or hardware, and the software performance overhead is minimal, and comparable to that required for NAT forwarding.

5 TRIAD Directory Services

In TRIAD, directory services are integrated into the relay agent nodes to match the availability of directory services with that of the basic communication layer. That is, if you can communicate with it, you can name it. Consequently, the TRIAD dependence on directory services does not reduce availability compared to the original Internet architecture. Integrating directory services into the relay agent nodes also means that the naming capacity is automatically upgraded as the communication capacity at a relay agent is upgraded

Each TRIAD relay agent acts as a name server for each realm to which it is directly connected. For names in the same realm as the requester, the TRIAD directory service behaves exactly the same as current DNS for current IPv4 clients making address requests. That is, a DNS request with $QTYPE = A$ simply returns the IPv4 address of the associated local host, as determined by the local name database. In particular, a relay agent can use name lookup to locate other RAs in the same ISP (and thus presumably connected to the same ISP realm.)

For inter-realm lookups, the relay agent also stores routing information, which can be statically configured, obtained through a dynamic routing protocol, or determined on an as-needed basis. In any case, some knowledge of where in the global network a name is located is necessary to generate IRTs for remote destinations. (Like *split DNS* in NAT, different realms will have different IRTs associated with the same name.) A relay agent which depends on a directory service to supply it with this information is called a *resolver relay*, while a relay agent which participates in a routing protocol is a *routing relay.*

The directory lookup relaying is supported by the *Directory Relay Protocol (DRP)*[4]. A name request for an endpoint in a remote realm is logically relayed along the path that packets are to take, based on local knowledge of which peer is the "next hop" towards an authoritative server for the name being requested. After the request reaches an authoritative server, a response is returned along the reverse path through the relay agents, with each one modifying the IRT to finally produce the IRT that the requester will use. (With transparent tokens, it just appends its address at the front of the IRP.) This behavior was illustrated earlier in Fig. 1.

By relaying the name lookup request across the same relay path as the packets are to flow, any necessary forwarding state can be set up in intermediate relay agents to handle the resulting IRT specification. Combining the path for the lookup with the path taken by the data packets ensures that each can be equally trusted or not. Relay agents also provide reverse (i.e., address-to-name) lookup by forwarding a reverse lookup request along the same path as a packet with the same address.

The relay nodes necessarily provide this directory service because addresses are not stable or meaningful between address realms. A node other than a relay node cannot reliably communicate with directory services outside its realm because IRTs relative to one relay agent are not meaningful when sent to a different relay agent.

[6] The test machine was a 333 MHz Celeron with 128 MB of RAM, running Linux 2.2.13.

In a multi-homed realm, such as an enterprise network served by two ISPs, the internal naming and routing selects one of the relay nodes for the name lookup based on local routing information. This mechanism is also expected to detect when a relay agent fails or becomes disconnected, causing traffic to be rerouted through the other relay agent. Because the name is the primary identifier and can be rebound without losing the connection state, the connection can survive this redirection to the other relay agent similar to a connection surviving rerouting in the current Internet. With routing updates significantly damped in the Internet to avoid oscillations, especially at the BGP level, we expect TRIAD name rebinding to provide recovery latency that is comparable to that of the current Internet.

5.1 Security

The directory service supports message authentication using public-key and shared-key cryptographic signatures. This allows clients to determine that the answer they get from the directory service is authentic, and allows relay agents to identify a particular principal associated with a client.

Unlike DNS security[5], a single name-to-address mapping cannot be signed by the authoritative server for a name because the address also depends on the intervening relay agents. Instead, relay agents must establish trust relationships.

5.2 Mobility

For mobile operation in TRIAD, a host visiting a guest network receives a temporary visitor name in that network (in a DNS domain of the visited network) which allows it to then communicate with the rest of the Internet. If the host needs to be reachable or authenticated as its normal DNS name, it communicates with its home directory service to insert a redirect specifying its current temporary visitor name in the guest network. When another host attempts to contact the visitor by its normal name, the home directory provides a redirect to the temporary visitor name, causing the other host to contact the mobile host in the guest network.

When the mobile host moves, it may keep connection state because it is associated with the endpoint names. It simply repeats the above steps in the new realm. With a non-cachable alias, (re)lookup of the normal DNS name of the mobile host provides a redirect to the new guest network. For real-time hand-off between guest networks, the mobile host requests the relay agent to forward packets addressed to its old address (using encapsulation) to the new guest network. This forwarding is canceled before the relay agent reuses the address and name that was used by this guest host (allowing the relay agent to use common state and time-out mechanisms to control the forwarding and the reuse). A reverse name lookup can also return the "real" name that redirects to this (temporary) name, providing what might be called *reverse aliasing*. A lookup on this real name is used to validate this reverse alias.

With this approach, the key mechanism to support mobility is the adding and removing of redirects in the home directory of the mobile host and the registering of aliases in the guest realm. The guest network simply needs to allocate and reclaim temporary addresses and names the same as supported by current DHCP services. It does not require routing all packets to a mobile host through the home gateway for the mobile host or encapsulating traffic to and from the mobile host, as mobile IP proposals imply.

5.3 Name-based Routing

At the inter-realm level, routing relays use *name-based routing* because addresses are not a meaningful way to identify endpoints in the wide area, and to ensure the availability of those names. (Intra-realm routing can use existing routing protocols, and intra-realm reliability of name service is ensured by duplicate servers as now, or possibly by router-integrated directory services.) Routing information is distributed among relay nodes and maintained locally with next hops and destinations specified in terms of names and name suffixes. With this step, the name directory and the routing table logically become a single entity, reducing the overall complexity of the directory and relay agent software. Note that a conventional routing table is a simple directory: It is queried with an IP address to determine the forwarding information. With TRIAD, the equivalent directory in a relay node is queried using the DNS name.

The *Name-Based Routing Protocol (NBRP)* [2] performs routing by name with a structure similar to BGP. Just as BGP distributes address prefix reachability information among autonomous systems, NBRP distributes *name suffix* reachability among address realms. NBRP updates are authenticated by cryptographically signing "delegations" of part of the namespace to a relay agent's peers, in a manner similar to

Secure BGP [16].

The key challenge with a name-based routing is maintaining the routing database efficiently even in the presence of names that do not match the routing hierarchy. Two mechanisms in NBRP reduce routing table size to a feasible level. The redirection mechanism explained above for mobility handles hosts whose names do not match network topology. (For example, all hosts with Harvard names not in the same address realm as the authoritative server for `Harvard.EDU` would have redirect records at that server.)

The other important component of NBRP is support for combining collections of name suffixes into *routing aggregates*, so that routing updates typically update a small number of aggregates rather than many individual entries. All names in a routing aggregate may be treated identically in routing calculations, thus reducing load at relay agents. Typically, we expect an ISP relay node to group all of the names from its customers into a small number of aggregates. Aggregate membership should be relatively long-lived, so that relay nodes can amortize the cost of learning the aggregate membership over many routing updates.

Another issue is keeping the number of neighbors small so that the routing overhead is acceptable. This can be accomplished by adding additional nodes on the interior of a realm which perform only route updates and name lookups, but not relaying. Current BGP speakers could be upgraded to perform as NBRP speakers as well, without necessarily acting as routing relays. As a degenerate example of the same idea, a typical ISP customer's relay agent may simply cache name and route information maintained by a relay agent (or its surrogate) located in the ISP's domain, eliminating the need for peering among ISP customers and reducing memory requirements for customer relay nodes.

5.4 Implementation and Evaluation

We have developed a prototype implementation of the extended directory and routing service required in TRIAD. The key issues are the client name lookup performance and the directory/routing storage and maintenance overhead.

Regarding name lookup, we expect most environments to use transparent IRTs, which have the *concatenation property* mentioned in Section 4. Thus, the caching of names behaves the same as with current DNS because a name server can lookup the address to an authoritative server once, then send name requests using the relay fast path to this server rather than through the name service on each intervening relay agent. The concatenation property also allows addresses looked up for one client to be used for another client on the same "side" of a relay node. Caching of names thus behaves the same as with current DNS.

Opaquing the IRTs can defeat caching, particularly if the returned IRT encodes source dependencies, but the cost is low compared to the other overheads with secure connection setup.

On a name cache miss, in TRIAD, the name lookup may proceed through several relay nodes, causing a full name lookup at each relay node. In contrast, a conventional DNS name cache miss (within an enterprise) causes a DNS request to be sent to a root name server. Thus, TRIAD may use more cycles in total, summed across several relay nodes, but it distributes this load over the relay nodes on the path of communication. In contrast, DNS incurs fewer total lookup cycles but concentrates the demand on the smaller number of root servers.

The number of name suffixes which must be searched is large, but not unacceptably so. There are currently 1.7 million second-level names in use world-wide, e.g. Harvard.EDU, Ietf.ORG, etc. (This number closely matches the number of suffixes obtained from the experiment explained below.) Assuming 64 bytes of space per entry (including hash indexes, etc.), storing the whole name database would cost 128 megabytes, an insignificant amount of disk space, even if the number was to be 10 times as much by the time TRIAD was deployed. NBRP table lookup is not on the packet forwarding fast path, unlike IP routing, so time spent searching the table is typically only paid during connection setup rather than per-packet. Note also that name lookups already encounter the cost of searching through a database of this size in conventional DNS.

In sum, we expect TRIAD name lookup to have comparable performance and scaling as current DNS, differing primarily for portions of the Internet configured for greater security requirements than supported by current DNS.

Considering the directory and routing overhead, at the ISP level, the name aggregation generally closely matches the address and routing aggregation. For example, Harvard.EDU corresponds to a small number of IP address ranges that further correspond to a small number of routes. This strong correspondence means the aggregation feasible with routing table entries is largely intact in going to name-based routing and

directory services. Conversely, organizations with large numbers of hosts scattered throughout the Internet are uncommon.

To evaluate the expected performance of name-based routing in the current Internet, we processed a comprehensive list of address-to-name mappings in the Domain Name System[17] and BGP table dumps from the MAE-East exchange point[18] by the following algorithm, making the assumption that address realm boundaries roughly correspond to current BGP autonomous system boundaries:

1. Each address range from the BGP table is matched with the DNS zones represented. (If fewer than *site_threshold* hosts in a range belong to an existing zone, they are removed from the table completely and assumed to be handled with the redirection mechanism.)

2. Names whose associated routing information is made redundant by a superzone are also removed.

3. Aggregates are created for any set of names larger than *aggregate_threshold* that have identical routing information (i.e., all known routes were identical, not just the preferred route.)

The resulting aggregates match those expected to be generated in a TRIAD relay node.

One representative set of results is shown in Table 1. These numbers indicate that NBRP results in a

site_threshold	Affixes (1000s)	*aggregate_threshold* 3	5	10	20
2	1727	19.5 (6.7)	20.1 (5.6)	25.7 (4.4)	37.0 (3.4)
3	1692	14.9 (5.9)	16.1 (5.0)	20.6 (4.0)	30.1 (3.2)
10	1679	14.8 (5.9)	16.0 (5.0)	20.6 (4.0)	30.3 (3.2)
original BGP	68.2	11.8			

Table 1: Number of routes (and aggregates) in thousands for different site and aggregate threshold values. With a site threshold of 10 and an aggregate threshold of 3, NBRP produces approximately 14,800 routing table entries (and 5,900 aggregates) which improves significantly on the original BGP number of 68,200 routing table entries.

set of destinations (and thus update frequency) comparable to BGP; higher-level aggregation may be able to reduce this yet further without resorting to renumbering or renaming.

BGP does have a limited mechanism for aggregation: a single route update may include several address prefixes. It is not clear the extent to which BGP software makes use of this to optimize update calculations: there is no requirement that advertisements keep these address prefixes together, and the address ranges must appear separately in the IP routing table. The entry in Table 1 corresponding to "original BGP" with an aggregation threshold of 3 indicating 11,800 entries indicates the best possible number of routes with BGP aggregation.

Addition of a new name is common, unlike addition of new BGP prefixes, and this name information must propagate to all relay points. However, name addition is done on human time scales; during the recent past, third-level domain names have been added at about 12 per minute. To put this in perspective, a backbone router may receive more than 2,000 routing updates per minute. Also, the actual level of routing updates necessary for new names will be lower, since changes to aggregates can be "batched" to reflect many new names with one update.

6 WRAPID Gateways

A WRAPID (WRAP-to-IP-Domain) gateway allows existing IPv4 end hosts to benefit from TRIAD without modifications to their software. WRAPID provides translation between IPv4 addresses and WRAP addressing similar to the IPv4 to IPv4 translation provided by NAT boxes.

When a given IPv4 host attempts to communicate with a remote host (outside the current address realm), the WRAPID gateway allocates an IPv4 address and sets up a mapping to the appropriate WRAP header. This header may map directly to the remote host or to a WRAPID gateway that serves that host. When an (IPv4) packet is sent to this allocated address, the WRAPID gateway translates the packet to a WRAP packet with the appropriate header and forwards it onwards. On receiving a WRAP packet from an external

host, the WRAPID gateway translates the packet to a simple IPv4 packet with the IP source appearing as this locally allocated address.

The WRAPID gateway can also implement WRAPsec, providing secure communication to the other WRAP endpoint, either a WRAP-enabled host or another WRAPID gateway.

The WRAPID gateway allows WRAP to be deployed incrementally, without being gated by end host adoption. In particular, one can have hosts on the same subnet being WRAP-enabled while others are not, yet still able to communicate with each other as well as hosts in other address realms. The optimization of eliminating the WRAP header when communicating within the same address realm means that a WRAP-enabled host never sends WRAP packets to other hosts in the same realm, so there is no need to discriminate between these hosts as part of local communication. Only the directory service interfacing to the rest of the Internet needs to distinguish. However, native WRAP (with the attendant host changes) is required to provide end-to-end security and reliability.

7 TRIAD Deployment

TRIAD has a clear, low-cost deployment path, based on user need, allowing TRIAD to be realized as an incremental evolution of the current Internet.

Consider as an extreme example a country with limited IPv4 addresses such as Thailand. The limitations of conventional NAT make it questionable as a solution to providing more addresses yet moving to IPv6 does not make sense either (see Appendix). However, with TRIAD, each such country can install a WRAP relay router that interfaces to the Internet. Attached to this top-level relay are one or more WRAPID gateways that include conventional NAT capability. The conventional NAT capability allows these hosts to communicate with the existing conventional IPv4 Internet. Each ISP, country or even organization that adopts TRIAD is able to communicate with other organizations using TRIAD *without* consuming any of its precious and limited global IPv4 addresses[7]. For instance, if Thailand and Indonesia both adopt TRIAD, they then have virtually unlimited addresses internally and between themselves, and are only constrained on the number of addresses they have available to communicate with the current Internet. (This is actually the same situation as if they had internally converted to IPv6, given they would still have to communicate with the rest of the planet using IPv4. But, with IPv6, they would also have to upgrade all their existing hosts and networking infrastructure.)

Thus, each organization is motivated to adopt TRIAD because it allows them to communicate with other TRIAD organizations without using their limited global IPv4 addresses, and because it makes it easier for other TRIAD users to communicate with them. So, those organizations that are currently short of addresses are motivated to move to TRIAD and those that are not are still motivated if they are interested in having the former communicate with them. Given that most of the major web sites are in the United States, and the U.S. companies have been in the lead to build Web-based operations, there would be considerable commercial motivation to support TRIAD in the American web sites once service providers in other countries were using TRIAD among themselves.

This initial deployment requires no real changes to end hosts and no change to the basic IPv4 routers and switches constituting the infrastructure of the leaf and backbone networks. It only requires the deployment of relay agents and WRAPID gateways, but these are modest extensions of the current NAT-enabled routers. Here, we assume that end-user applications have been or will be modified in any case to deal with the lack of meaning of addresses across NAT boundaries.

Once WRAP is deployed to some degree in the Internet, first host implementations are expected to arise with large-scale servers where eliminating the extra overhead, delay and point of failure of a WRAPID gateway may be warranted. Making an externally accessed server WRAP-enabled also eliminates the server use of an externally visible (IPv4) address which, with an active server, would be essentially allocated indefinitely to this server. During this transition, conventional IPv4 hosts and TRIAD-aware hosts can easily and efficiently co-exist in the same address realm. Given that WRAP appears relatively straight forward to implement, the main delay in getting all hosts upgraded to WRAP is expected to be the basic inertia in getting changes into commercial software and getting administrators of systems to upgrade their software. Hosts that need end-to-end security and reliability are also motivated to upgrade to native WRAP.

[7] This assumes the gateway already has one such address if the WRAP RAs communicate over the existing wide-area IPv4 infrastructure.

NBRP is just one means of providing the name-to-authoritative server mapping needed by DRP. This same information already exists in the current Domain Name System (in the form of NS records, which organizations with domain names already control) so resolver relay agents can make use of the existing naming infrastructure to locate other TRIAD gateways rather than participating in a dynamic routing protocol. We expect deployment to occur mainly at the edges of the network, and thus cannot depend on ISPs providing new infrastructure. Such a scenario could lead to an topology that would have many thousands of realms peering in the "global" Internet, but there is little need for NBRP in such an environment. The amount of topological change in such a situation is small, and multihomed sites can easily list all their gateways as NS records rather than constantly updating the DNS. Later, these resolver relays can be upgraded to routing relays as ISPs begin offering NBRP.

8 Advantages of TRIAD

TRIAD provides end-to-end semantics across NAT boundaries, allowing for secure and reliable communication while providing all the benefits of NAT. As mentioned in the introduction, NAT is valuable for address assignment autonomy, flexible multi-homing, concealing the internal addressing of an organization, and transparent redirect as with virtual hosts.

TRIAD extends IP addressing, allowing the Internet to scale arbitrarily without having to make the painful transition to IPv6[8]. This extended addressing also allows TRIAD to support VPNs without having to upgrade all routers to support MPLS or incur the full overhead of tunneling. Similarly, it supports policy-based routing across realms and the extended forwarding path check as a scalable extension of the RPF check.

TRIAD provides a reliable integrated directory service, allowing all identification to be based on user-assigned names without compromising reliability. These names are used for authentication, making what is used by applications and what is secured the same, in contrast to IPsec. In addition, the TRIAD directory service supports multicast channel naming, mobility, policy-based routing and DNS-level load balancing, removing the complexity of these facilities from the network layer.

TRIAD provides trusted reverse name lookup, at least to the extent the receiver can trust the packet.

TRIAD incurs a lower space and time overhead for communication on average because communication within a realm just uses the conventional IPv4 header. Given that most communication is local and the current Internet with NAT boxes is effectively at most 3 relay agents to anywhere, the packet header overhead on average is expected to be significantly less with WRAP than with IPv6. This header overhead is significant because most packets are small and per-packet processing is a significant cost with small packets. This optimized local communication also suits small embedded systems, many of which use or will use limited bandwidth wireless communication. Moreover, it is readily hardware implementable because of the size of relay address to lookup can be fixed size.

Finally, TRIAD is readily deployable incrementally as outlined in the previous section. There is no need to change the network infrastructure within an address realm or to change backbone routers and management. The boundary (NAT) routers are upgraded to support TRIAD and then the hosts can then be individually upgraded to use WRAP natively.

9 Related Work

NAT seems to have been introduced into the Internet in Jacobson's insightful early proposal [9] although the same techniques appear in some earlier distributed systems work [8]. Since 1992, various RFC's [10] have clarified the use of NAT, provided for private addresses [11] and clarified the terminology, use and problems [15]. Industry has deployed a variety of products supporting network address translation, including firewalls, routers and server load balancers.

More recently, work on IPsec and others have recognized the problems with basing identity on IP addresses and the conflict of end-to-end security with the increasing deployment of NAT.

RSIP [13] is an alternative approach to dealing with NAT, where a host in a NAT realm explicitly obtains an external IP address, tunnels packets through the NAT gateway using this external IP address and thus can

[8]One could argue that the Internet does not actually need more global addresses, by relying on efficient allocation and NAPT, given only about 1 percent of the IPv4 addresses are actually in use. However, this argument makes IPv6 even less compelling and WRAP is still beneficial for other reasons, such as connecting private address domains.

use IPsec and other protocols without requiring NAT translation. However, RSIP requires host modification to operate in this mode and it does not increase the number of external IP addresses. With all the extra benefits that TRIAD provides, it seems more effective and lower risk to modify the hosts to support native TRIAD, incrementally and once TRIAD has been widely deployed between realms.

The WRAP relay model we use, as an extension of the basic forwarding level of conventional routers, appears unique as we have defined it. WRAP relaying is similar to loose source routing except the packet is forwarded at each relay agent with the source IP address of the packet being one set by the relay agent, not the original source address. Moreover, each specified address is in a separate address realm, with translation between address realms occurring at each realm boundary. Source routing provides a source-controlled path but does not cross realms and does not change the source address on each hop, as is required for inter-realm communication. It is also similar to tunneling except the relay path taken by the packet (analogous to the sequence of one or more tunnels) is effectively recorded in the packet and, as above, translation of addresses may take place. The relay approach also uses a far smaller header than a nested set of IP headers, as required for a multi-tunnel path.

IP tunneling has been used to effectively extend addressing by tunneling from one realm to another. However, tunneling makes layer 4 filtering harder because, with multi-hop tunneling, the location of the layer 4 header involves parsing each encapsulation. Also, unlike WRAP, the path the packet takes is lost with tunneling. Moreover, tunneling incurs greater overhead than WRAP and requires that the source know the path. Moreover, the packet size does not change with WRAP, unlike encapsulation and de-encapsulation that occurs with tunneling.

MPLS [12] provides tagging of packets similar to WRAP, but below the IP level. MPLS does not provide more addresses beyond that provided by NAT, unlike WRAP. On the other hand, WRAP can be used intra-realm and inter-realm for traffic engineering and VPNs, reducing, if not eliminating, the need for MPLS[9]. MPLS also requires special support in the forwarding path of *all* routers on the path, whereas WRAP/TRIAD only requires support at the border or relay agents. MPLS also requires a new mechanism for distributing tags. MPLS does not save the path a packet followed either. While the WRAP header does impose a higher overhead than an MPLS tag, it is less than IPv6 and less than conventional IPv4 tunneling, especially with multi-path tunnels. Thus, IP4 plus MPLS is not a solution to scaling and IPv6 plus MPLS carries all the disadvantages of IPv6 plus the MPLS overhead. Both WRAP and MPLS make the offset of the TCP/UDP ports variable within the packet, affecting the design of access control filters on packets. However, with the length field in the WRAP header at a fixed offset, it is straightforward for even a hardware implementation to determine the actual offset of layer 4 ports, as required for access control processing. Moreover, in initial deployment, we expect that firewalls may simply restrict WRAP packets to specific WRAP-enabled hosts, such as WRAPID gateways, which can filter further as needed.

Recent IETF work has promoted "transparency" as an important property to achieve in the Internet, defined as "a single universal logical addressing scheme and the mechanisms by which packets may flow from source to destination essentially unaltered" [14]. We view that TRIAD provides transparency under this definition, viewing the "logical addressing scheme" to be DNS naming and the transmission of data without changing the data or its checksum as "essentially unaltered". The changing of the addressing in the packet is not real alteration because corruption by intermediate points is as detectable as with conventional end-to-end delivery.

10 Future Work

Our current work is focused on designing and implementing the detailed protocols required by TRIAD if it is to be deployed in the Internet, and performing further evaluation and study to support this direction. Specifically, we expect to perform much more comprehensive studies of the large-scale behavior of name-based TCP and routing through simulation. The ability of name rebinding to handle routing topology changes efficiently and the effects of route aggregation need to be clearly demonstrated before TRIAD can achieve wide-scale deployment.

The performance of relay nodes also merits further investigation, especially demonstrating hardware that performs WRAP relaying at the necessary speeds to deploy at ISP boundaries. The effects of opaque

[9] One of the original motivations for MPLS, efficient IP forwarding, has been eliminated by the advent of wire-speed hardware IPv4 forwarding engines.

tokens on relaying performance and their additional costs (such as potential loss of cachability) are two other performance issues.

However, we have confidence in TRIAD's scalability, since the dynamics of naming and routing are similar to what already exists in the IPv4 Internet. Futher, the incremental deployment of TRIAD gives us the opportunity to tune and refine protocols futher as they are being implemented and deployed in the network.

11 Concluding Remarks

Scaling the Internet is a multi-dimensional problem, spanning not just the number of nodes, but also the capabilities of those nodes, the diversity of security and protection demands, the diversity of available bandwidth, and the complexity of network topology. A new architecture must recognize the multiplicity of demands, from small embedded systems to very large distributed organizations, across a wide range of administrative boundaries and requirements. We believe TRIAD is a promising new architecture on which to base the future of the Internet, which addresses all of these issues, not just the low-level problem of too few network identifiers.

TRIAD solves the problems of NAT, allowing it to be accepted as a legitimate part of the Internet architecture, and sensibly deployed for its many uses. TRIAD also provides an significantly improved directory service, ensuring better naming support for applications and allowing multicast naming, mobility, policy-based routing and wide-area load balancing to be implemented at the higher level, rather than complicating the network layer. Moreover, TRIAD is largely IPv4 and DNS compatible for end hosts, leaf and backbone networks, simply requiring extensions to NAT-capable boundary routers. Thus, it retains the key aspects that have allowed the Internet to be so successful to date, including end-to-end semantics, and addresses one of its major deficiencies, namely lack of a dependable directory service.

Compared to IPv6, TRIAD is more backwards compatible, more deployable, more efficient and more secure while providing the same end-to-end semantics and recovery relative to network failures.

TRIAD, as the name suggests, is based on three key ideas. For one, TRIAD makes the user-assigned host or multicast channel DNS name the only global identifier, making packet addressing local and transient, and thereby short, efficient and automatically assignable. It integrates the directory system into the network infrastructure of routers to ensure availability and trust that matches the network itself.

Second, TRIAD uses a name-based packet pseudo-header, a natural approach, given the replacement of addresses by names in general. This approach supports end-to-end semantics even though the packet addresses are translated at each relay agent. TRIAD-TCP also uses name-based connections and name remapping as part of recovery to make the translation and relaying state in the network "soft", providing the same end-to-end resilience to failure as the original Internet architecture.

Finally, TRIAD introduces a relay layer and a shim protocol WRAP between IPv4 and transport protocols, providing extensible addressability between address realms and independence of addressing within each realm. In particular, local address structure can be completely hidden from the rest of the Internet and external Internet structure can be completely hidden from the local realm. Moreover, intra-realm communication can optimize out WRAP, incurring the same packet overhead in size and processing as IPv4. The simplicity of WRAP makes it feasible to implement in hardware in the next generation of switch/routers, allowing wire speed relaying, even at the highest performance levels.

A broader contribution of this work is the recognition of the value of NAT and directory services to the Internet. Both are critical and integral aspects in practice, yet both were omitted from the original architecture. They clearly need to be incorporated as we have done in TRIAD if there is to be an architecture that matches reality. In particular, the view of NAT as an interim hack is just plain wrong.

We believe that the primary competition to TRIAD at this stage is the continued *ad hoc* deployment of NAT and application-level proxies, not IPv6. Continued growth of the Internet without a guiding architecture will significantly detract from its reliability, security and eventually, utility.

References

[1] David R. Cheriton and Chetan Rai, Wide-area Relay Addressing Protocol (WRAP), in preparation. 1999.

[2] Mark Gritter and David Cheriton, Name-based Routing Protocol Specification, in progress, 1999

[3] Hugh Holbrook and David R. Cheriton, IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications, Proc. ACM SigComm'99 Cambridge, MA Sept. 1999.

[4] M. Gritter, Name-enabled Routing and Directory Services in TRIAD, in progress, 2000.

[5] D. Eastlake, Domain Name Security Extensions, RFC 2535, March 1999.

[6] P. Ferguson, H. Berkowitz, Renumbering: What is it and why do I want it anyway, RFC 2071, January 1997.

[7] S. Deering and R. Hinden, IP Version 6 Addressing Architecture, RFC 2372, July 1998.

[8] D.R. Cheriton, Local Networking and Internetworking in the V-System, Proc. 8th Data Communication Symposium, IEEE/ACM, 1983

[9] V. Jacobson, LNAT — Large-scale IP via Network Address Translation, working draft, Lawrence Berkeley Labs, Jan. 1992.

[10] E. Egevang and P. Francis, The IP Network Address Translator (NAT) RFC 1631, May 1994.

[11] Y. Rekhter, B. Moskowitz, D. Karrenberg and G. de Groot, Address Allocation for Private Internets, RFC 1597, March 1994.

[12] E. Rosen, A. Viswanathan and R. Callon, Multi-protocol Label Switching Architecture, work-in-progress, Internet draft, 1999.

[13] M. Borella, D. Grabelsky, J. Lo, Realm Specific IP: Protocol Specification, draft-ieft-nat-rsip-protocol-03.txt, Oct. 1999 (work in progress).

[14] M. Kaat, Overview of IAB 1999 Network Layer Workshop, draft-ietf-iab-ntwlyrws-over-01.txt, work in progress, Oct. 1999

[15] NAT Working Group, P. Srisuresh and Matt Holdrege, IP Network Address Translator (NAT) Terminology and Considerations draft-ietf-nat-terminology-00.txt, work in progress, July 1998

[16] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)", to appear in IEEE Journal on Selected Areas in Communication, 1999.

[17] Internet Domain Survey, July 1999, `http://www.isc.org/ds/`.

[18] Internet Performance Measurement and Analysis project, `http://www.merit.edu/ipma/`.

[19] Wireless Application Protocol Forum `http://www.wapforum.org`.

A IPv6 Deployment Challenges and Risks

IPv6 deployment faces a number of challenges, including: 1) the IPv6 costs and risks, 2) the fact that NAT is required to incrementally deploy IPv6 yet appears to eliminates the need for IPv6, and 3) the inability to really use the IPv6 features effectively during incremental deployment. This section considers possible initial deployment scenarios to illustrate the difficulties and then discuss some further risks.

One view is that IPv6 should be first deployed at the enterprise level. However, to deploy IPv6 in some portion of the corporate network would require network address translation between the IPv6 portion and the "legacy" IPv4 portion, including the rest of the Internet. However, given network address translation, it would be lower risk to instead deploy more IPv4 using a private address domain and thereby gain sufficient addresses for the immediate enterprise needs. This route would eliminate the risks of disrupting the end hosts, routers and multi-layer switches, and network management systems to upgrade to IPv6. NAT-based solutions are widely deployed and well-understood whereas IPv6 support is still largely experimental. Thus, it seems difficult to get IPv6 deployed initially in an enterprise network.

Another view is that IPv6 should be deployed first in the backbone of the Internet. Yet, this appears to expose the ISP to unjustified costs and risks. The backbone has relatively few nodes so it does not have the

demand for addresses to compel going to IPv6. Moreover, the ISP would have to support both IPv4 and IPv6 unless its customers simultaneously convert. (Dual-stack mechanisms and tunneling consume extra network and human resources over supporting just IPv4.) Finally, the ISP would have to provide backbone routers with adequate performance. However, there is no existing market for such products and relatively little investment in this direction because there is no significant amount of IPv6 traffic. So, it is not clear where and when an ISP would get these routers from even it decided to convert to IPv6.

Yet another view is that IPv6 might be widely deployed by some wireless service such as cellular phones. However, this move would incur higher packet overhead unless header compression can be very effective[10]. Also, the average packet size with wireless tends to be smaller, both because the technology and because voice uses small packets. Moreover, a key challenge with wireless is dealing with many units collecting in the same cell, whether they are cell phones, wireless appliances in the home or other wireless mobile devices. If IPv6 does increase the packet overhead significantly, it effectively reduces the maximum number of units that can be served per cell in the worst case, thus increasing the cost. Moreover, wireless only has to transmit to the nearest (wired) receiver, which offers an opportunity to translate the packets to another format for the wired infrastructure, as proposed in the widely supported WAP standard [19]. The arguments for IPv6 based on autoconfiguration may also be less compelling given that wireless devices have to authenticate themselves when entering a realm, giving ample mechanism and opportunity to do DHCP-like address assignment at that point. Finally, having fixed IPv6 addresses for mobile hosts is mostly interesting to support mobile IP, yet mobile IP has received relatively little deployment to date, given the routing difficulties and solutions that exist at the higher level. Until mobile IP is more compelling and excessive header overhead is shown to not be an issue, or until another compelling reason is identified for IPv6 on cellular phones, it is hard to see IPv6 being deployed in this domain.

A final view is that some country such as China will so desperately need addresses that it would deploy IPv6. However, this scenario raises a number of issues. To communicate with the rest of the existing Internet, China would require sufficient (global) IPv4 addresses in any case for NAT-based communication from its internal IPv6 to these IPv4 addresses. However, if it has these addresses[11], it would be far easier to run the whole country behind a NAT boundary using IPv4 addresses, given that that would allow the use of existing routers, switches and host software. It seems inadvisable for a country with limited Internet expertise and industry to commit to the least proven technology and possibly be forced to largely develop its own products, especially with uncertain prospects of other markets for these products.

Besides all the above impediments to deployment, there are significant technical risks to deploying IPv6, in part because of the ancillary changes made compared to IPv4 besides just the change in address size. Although the IPv6 work has shown admirable restraint in avoiding gratuitous changes over IPv4, there are enough of these differences from IPv4 to have legitimate concern that unanticipated problems will arise, given that existing applications were designed, debugged and deployed up based on IPv4.

In particular, with IPv6, the address assignment in the high-order 64 bits is allocated to ISPs so, if an enterprise network is served by two ISPs (for fault-tolerance or choice of service), every IPv6 host in the enterprise is effectively multi-homed, with two separate addresses per host, one for each of the ISPs. If one of the ISP's service fails, the addresses used by those sending to this network have to change to those associated with the other ISP for fail-over to occur, unless one of the complex tunneling or rerouting schemes currently being researched to handle this problem can be made to work reliably, or some other solution is available.

IPv6 introduces a privacy risk because it encodes information in the addresses, making this information externally visible. For instance, with IPv6, one can determine a company's ISP based on the addresses used by its hosts. IPv6 also makes every host that uses multiple ISPs effectively multi-homed. IPv6 addresses can also encode MAC addresses that can reveal the manufacturers of the Ethernet interfaces in the hosts. These issues have already caught the attention of privacy groups.

IPv6 relies on "renumbering" [6] for efficient routing to keep the mapping of address to topology reasonably compatible. It is reasonably considered a research issue because there is no prior system to the authors' knowledge that has proven this is in fact practical.

[10] Conventional header compression works best on long-lived connections, such as telnet sessions over dialup links. However, cellular phone data services have been most successful with short message services, for which the gains are less clear.

[11] At the time of writing, China has approximately 7 million global addresses, enough to support approximately 500 billion simultaneous connections using NAT-PT.

IPv6 also changes the way that options and IP fragmentation are handled. In particular, IPv6 disallows fragmentation at intermediate hops, making it even more difficult to use multicast efficiently in a highly diverse environment. Some networks impose fragmentation on large packets to provide delay guarantees for latency-sensitive traffic. This fragmentation may only come into play when such applications are running. It seems inappropriate to force a small MTU on a distant multicast source, for all receivers, just because a local low bandwidth link is carrying voice, for instance.

The large IPv6 header also introduces significant overhead and risk in some network settings. Besides the overhead in low bandwidth settings and/or risk that header compression will not be effective, the larger header may cause some applications with fixed packet sizes, like those tuned to Ethernet maximum packet size, to incur fragmentation at the IP level because of the larger header, a further deployment risk. IPv6 requires extensive changes to existing end-user host software and the network infrastructure of routers, switches, firewalls and network management. This IPv6 software and equipment is far less tested, less well-supported and far less cost-effective than the comparable IPv4 facilities.

Furthermore, routers are making a rapid transition to hardware support for IPv4 wire-speed forwarding, especially for core or backbone routers. There is the risk that IPv6 hardware support will be lagging and far more expensive, leading to substantially lower performance and/or much higher cost[12].

Finally, early adopters risk being orphaned if IPv6 is not be widely deployed soon after they make the move, incurring the cost of backing out of IPv6 as well as the risks and costs of conversion. The lack of IPv6 deployment to date provides empirical support to the above concerns.

Given mission-critical nature of networks and the rapid growth of traffic that most networks are confronting, few can afford to take on the IPv6 challenges and risks at this time.

[12]For instance, a modern hardware router can forward IPv4 packets at 50 million packet per second yet only forward IPv6 in software at roughly 50 thousand packets a second.