An Internet without the Internet Protocol

Craig A. Shue and Minaxi Gupta
Computer Science Department, Indiana University
{cshue, minaxi}@cs.indiana.edu

Abstract

The growth of the Internet has brought about many challenges for its critical infrastructure. The DNS infrastructure, which translates mnemonic host names into IP addresses understood by the routers, is frequently the target of cache poisoning attacks. Internet routers are also experiencing alarming growth in their routing table sizes, which may soon make it impossible for them to forward packets quickly enough to meet demand. Further, concerns about IPv4 address space exhaustion loom on the horizon despite the availability of IPv6. In this paper, we take a fresh look at Internet routing and propose a scheme that addresses all of these concerns cleanly. Our scheme forgoes IP addresses entirely and instead uses host names as *identifiers* in packets. The scalability of routing is ensured by encapsulating these packets in highly aggregated routing *locators*: we use autonomous system numbers (ASNs), which are already an integral part of inter-domain routing. We present data and experiments to show that a much simpler and scalable routing infrastructure can be designed for a future Internet by using fewer identifiers for its entities.

Key words: new Internet architecture, routing, host identification

1. Introduction

While the Internet appears to be functioning well, serious threats loom on the horizon. One reason for concern is the *scalability of routing*: The routing tables in the core of the Internet are growing at an alarming rate and many worry that these routers may soon be unable to meet demand [1]. To add to these woes are the ever-looming concerns about IP *address space exhaustion.* Although IPv6 [2] is widely accepted as a replacement for IPv4, its adoption has been far from stellar[1]. Further, IPv6 will not address some causes of growth in IPv4, such as multi-homing or load balancing. IPv6 forwarding also requires greater memory and more processing time than IPv4 forwarding [4], increasing load at routers.

We take a minimalist approach to addressing concerns about routing scalability and address space exhaustion. In doing so, we asked ourselves a range of questions. One pivotal question was: *What would an Internet without IP look like?* To our own surprise, the answer was: *Faster, expandable, and more scalable!* Two key observations led to this answer:

Minimalism in inter-domain routing: The border gateway protocol (BGP) uses two different identifiers to perform inter-domain routing: It announces the reachability of IP prefixes but the path to these prefixes is expressed in the form of autonomous system numbers (ASNs), which helps BGP avoid routing loops. The use of prefixes over ASNs to announce reachability has unfortunate consequences for routing scalability: the number of prefixes in the Internet is an order of magnitude bigger than the number of ASNs. Moreover, in 2008, the number of prefixes advertised has increased by about $50,000$ (approximately a 20% increase) while the number of ASNs grew by about $2,500$ (approximately a 9% increase) [5, 6][2]. This growth in prefixes is in part due to traffic engineering and multi-homing, practices that are likely to continue into the future. IPv4 address space exhaustion also fuels prefix growth, with many smaller prefixes being added to the routing tables. Consequently, *we forgo the use of IP prefixes in inter-domain routing and propose to use ASNs in routing announcements and during packet forwarding.* This simple idea has been proposed in various

[1]According to the Route Views Project [3], which provides border gateway protocol (BGP) routing tables from many vantage points in the Internet, the highest number of IPv6 prefix entries at any vantage point was only $1,311$ in July 2008.

[2]Previous work indicated ASNs growth rates exceed that of IP prefixes [7]. However, this trend does not hold today.

contexts [8, 9, 10, 11], yet has far-reaching consequences. First, it reduces the forwarding table sizes at the core Internet routers by an order of magnitude. Second, it allows fixed-length lookups during packet forwarding, which are at least an order of magnitude faster than the *longest prefix match* currently performed on IP prefixes. Third, in the current Internet, IP prefixes act as *locators* and IP addresses are *identifiers* for end hosts. This tangles routing and addressing. Accordingly, many proposals in the research community have converged on the belief that *locator-identifier separation* is essential to designing a scalable routing core for the next generation Internet [12]. Our idea makes ASNs into locators and leaves identifiers open to innovation, naturally decoupling the two.

Minimalism in end-host identification: Most hosts today have two identifiers associated with them: IP addresses and fully qualified domain names (for simplicity, we refer to the latter as *host names* or *names* subsequently). The host names are indispensable, since end users rely on their mnemonic nature. However, the value of IP addresses becomes unclear with the decoupling of routing and addressing. Consequently, *we propose to eliminate IP addresses as a global identifier, leaving host names as sole end-host identifier*. An instant outcome of this choice is that it solves address exhaustion concerns. Since host names can be up to 255 characters long [13], the scheme can accommodate more hosts than even allowed by 128-bit IPv6 addresses. Today, host names are translated into IP addresses by the DNS infrastructure, which the routers use to transmit packets. Under our scheme, users will continue to use host names but the DNS will be modified to translate them to ASNs, which the routers will use to transmit packets.

In designing *an Internet architecture without the Internet Protocol*, we leverage the recent proposals that have advocated using locators that differ from IP prefixes, including NIMROD [8], LISP [9], eFIT [10], and ENCAPS [11]. Since these proposals have extensively discussed the mechanics of using locators to scale inter-domain routing, we focus on the details of using ASNs as locators in inter-domain routing. Specifically, we examine how ASNs impact the factors that are currently causing super-linear growth in inter-domain forwarding tables, namely multi-homing, traffic engineering, and address fragmentation. We also examine the resulting routing tables sizes and packet forwarding speeds. We find that ASNs possess properties that can enhance packet forwarding speeds today as well as scale Internet routing for decades to come. Using host names as identifiers reduces the requirements for the DNS, aiding caching and reducing the viability of many cache poisoning attacks, while providing extensive identification space and reasonable forwarding performance within destination networks. With these results, we find that such an architecture has significant advantages.

The rest of this paper is structured as follows. In Section 2, we provide background material. In Section 3, we discuss the details of our architecture. In Sections 4 and 5, we evaluate ASNs as routing locators and host names as identifiers. Section 6 discusses practical considerations and open questions associated with the architecture. We review related work in Section 7 and conclude in Section 8.

2. Background

In this section, we provide some background on the causes of routing table growth in the Internet. We also outline the key features of the proposals that utilize a locator-identifier split to scale Internet routing.

2.1. Growth in Inter-domain Routing Tables

Routing table growth is an alarming trend on the Internet and has been analyzed by the community. Bu *et al.* examined the causes of BGP routing table growth and found four key factors: routers' *failure to aggregate* prefixes that can be aggregated, *address fragmentation*, *load balancing*, and *multi-homing* [14]. Routers failing to aggregate prefixes that can be aggregated can easily be eliminated by careful router configuration on the part of network operators. Address fragmentation is the result of IPv4 prefixes being insufficiently large: when an organization exhausts the address space available under their first prefix, they must request another for their remaining hosts. This second prefix is frequently disjoint from the first, preventing aggregation. As a result, these two prefixes must be advertised separately and two entries are stored in routing tables. Load balancing, a popular traffic engineering technique, also increases the number of prefixes in routing tables. To distribute the traffic arriving at the organization, the originating AS may simply divide a prefix into pieces and announce the pieces through different neighboring ASes. Since the path for each sub-prefix is different, each sub-prefix must be stored as a unique routing table entry, inflating growth. Finally, multi-homing also inflates the routing table size when provider-dependent address space is used. In this approach, a customer may multi-home and use address space obtained from one of its providers. The customer announces a sub-prefix obtained from one provider through each of its providers. Since this sub-prefix has different routing properties from the provider's prefix, it must be stored as a separate entry. When provider-independent address space is used, the prefix must already be announced separately, so multi-homing does not cause additional growth in this case.

2.2. Locator-Identifier Split Proposals

Each of NIMROD, LISP, eFIT, ENCAPS, ISLAY aim to scale Internet routing by using locators that may be different from IP prefixes. Though they differ in details, the basic idea behind each of them is to have the routers close to the sources encapsulate each packet in a special wrapper that contains the locators for the source and the destination. Routers in the core of the Internet will forward packets based only on these locators. This imparts scalability to routing, since locators are expected to be fewer in number and will lead to smaller routing tables. When such a packet reaches a router near the destination, the router will de-capsulate the outer layer and forward the original packet to the destination. To accomplish the mapping of end-host identifiers to locators, the proposals advocate using a database, which will be responsible for keeping the mapping information current. The work in APT [15] defines a mapping service for the eFIT architecture while the work in NERD [16] specifies a mapping database suite for LISP. While LISP allows several options for the mapping database, most architectures require that the database and updates be sent to each encapsulating router. In NIMROD [8], routers use IPv6 addresses with 32 bit locator addresses, allowing multiple locators to be specified in a single address. The work by Krioukov *et al.* [7] studied these mapping databases and questions the scalability of this approach. However, a pull-based mapping distribution system, like the DNS, may offer greater scalability. Accordingly, we do not believe architectures that split locators and identifiers should be quickly dismissed.

3. Architecture Details

3.1. An Example

We begin by describing our architecture through an example. Suppose a client, `host1.isp.com`, in ASN `1000`, wants to communicate with a server, `www.website.com`, in ASN `2000` (see Figure 1). The following sequence of steps will occur:

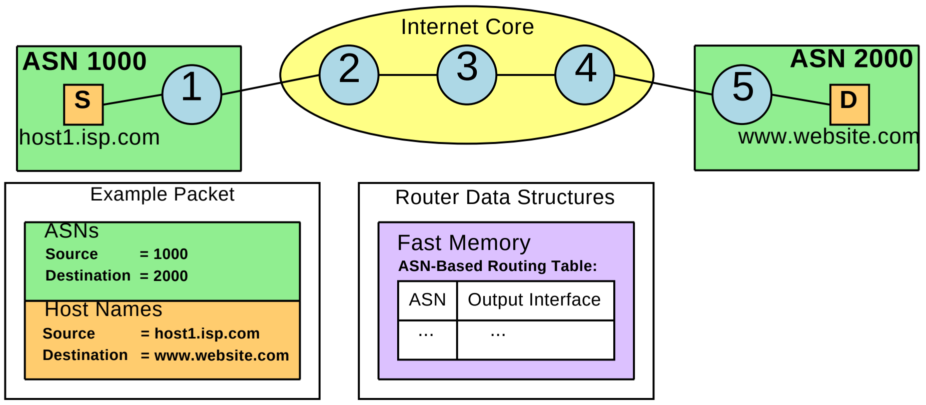

Figure 1: An example scenario

1. The client, which is configured with its own ASN information, contacts its DNS resolver to get the ASN for `www.website.com`. The client then creates a packet with a header containing `host1.isp.com` and `www.website.com` as source and destination addresses and `1000` and `2000` as source and destination ASN locators. The packet is sent to the default router, `1`.

2. Router `1` forwards the packet by looking up `2000` in its forwarding table, as prescribed by NIMROD, LISP, eFIT, and ENCAPS. These routers use an inter-domain routing protocol, such as a modified variant of BGP or HLP [17], that exchanges ASN reachability rather than prefix reachability. Subsequent core routers, routers `2`, `3`, and `4`, repeat the forwarding table lookup performed by router `2`.

3. When the packet arrives at router `5`, the router recognizes its own ASN locator and forwards the packet using the destination host name, `www.website.com`. A later optimization, introduced in Section 3.4, augments the host name with subnet identifiers and link layer information to expedite intra-domain forwarding; however, the simple name-based forwarding approach is sufficient for this illustration.

4. To reply to the client, the server simply reverses the source and destination host names and ASN locators.

Our architecture bears similarity to the design of the modern Internet but also differs from it in substantial ways. In our architecture, the clients contact the DNS resolvers only for the first packet of the connection, just as they do today. However, the DNS response packets return only ASN locators instead of IP addresses for end hosts. Also, the packet headers in our architecture contain source and destination host names and ASN locators while today they only contain IP addresses. Further, the routers forward packets based on ASNs, which bear little resemblance to the IP prefixes used today.

3.2. Leveraging Locator-Identifier Split Proposals

As described, our architecture leverages the various scalable-routing proposals, including NIMROD, LISP, eFIT, and ENCAPS, but requires a few changes to their functionality. Since these proposals implicitly assume that the end hosts may stay the same as today, they require that the routers close to the clients map end-host identifiers to locators. These locators are not part of the original packet header. Instead, they are placed in a wrapper that encapsulates the original packet. Routers in the core of the Internet only consult this wrapper while making forwarding decisions. When the packet reaches a router close to destination, the wrapper is stripped off and original packet retrieved. The original packet is then forwarded toward the destination using intra-domain routing, as it would today. However, our architecture requires a few fundamental changes to the scalable-routing proposals. First, locators are first-class citizens of the network layer in our architecture – the end hosts put this information in the packet by consulting an updated DNS-like database. This saves the routers from having to maintain the database that maps end-host identifiers to host names. It also saves the routers encapsulation and decapsulation effort. Second, these proposals leave the DNS unchanged and advocate using a separate database (residing at the routers in some cases) to maintain host identifier to router locator mappings. Rather than use a separate database, we evolve the DNS to hold these mappings since the DNS is no longer required in its original form under our architecture. Finally, different proposals endorse different types of routing locators. In our work, we show how ASNs can serve as these locators, providing an instantiation of these designs.

3.3. Header Design

Our architecture will have two components: a layer with inter-domain routing locators which will be used by NIMROD, LISP, eFIT, ENCAPS or other locator architectures and a layer for end-host identification which will be used by intra-domain routing protocols. The router locators are stored in a layer below the identification layer and that layer is dependent on which locator protocol is in use. This layered model allows the addressing layer to be utilized over diverse locator layer implementations.

Figure 2 shows a basic design of the new identification layer that contains variable length source and destination host names instead of IP addresses. We leverage the IPv6 header to arrive at this preliminary design. To make processing the variable length names easier, the header also contains the length of both the names. These 8 bit values indicate the number of characters present in the name. Notice that the length indicates the number of characters, not bytes. Due to the restricted character set of domain names, each character can be represented in 6 bits. Names that do not terminate on the byte boundary are padded with zeros to the next byte boundary.

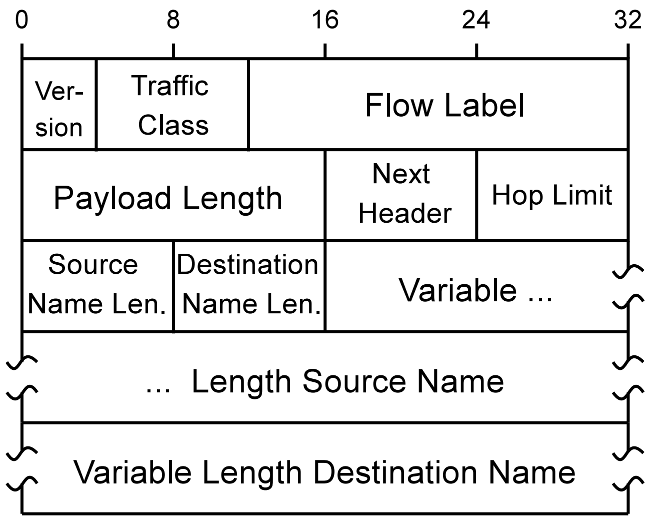

Figure 2: Header for the identification layer

3.4. Intra-domain Routing Optimizations

In the architecture described thus far, intra-domain routing is performed on host names. This approach is feasible for smaller ASes with few domains and host names that must be routed, as we show in Section 5.2. However, other ASes may have large networks and service many domains. In the case of Web hosting providers, a single IP address can provide hosting to over $10,000$ DNS host names [18]. In such ASes, routing on host names may require too much memory or forwarding time. Fortunately, other techniques can be used to minimize the forwarding state and lookup requirements.

In IP intra-domain routing, IP prefixes are used to direct a packet to the appropriate subnet, at which point the packet is forwarded using link layer information. In our approach, we can map the packets from their host names to their subnet and MAC addresses at an organizational name server. These packets can then be forwarded using the subnet identifier to the appropriate router, then forwarded using the supplied MAC address. To do so, the destination host name can be rewritten by the name server to include the subnet identifier and MAC address, allowing routers to quickly transmit packets while preserving the host name.

The host name rewriting approach can leverage the fixed character set of DNS host names. Host names can contain 38 valid characters (letters, numbers, periods and hyphens) and can be represented by a 6 bit value. However, this representation results in 26 unused 6 bit sequences. These unused sequences can be used to indicate whether a subnet ID and MAC are encoded in the host name, and if so, how many bits these values use. Accordingly, each AS can independently determine the number of subnets needed and arbitrarily encode these IDs and the MACs in the host names. End-hosts receiving packets would be able to detect and skip the subnet ID and MAC addresses and examine the host name portion to identify the communicating end-point.

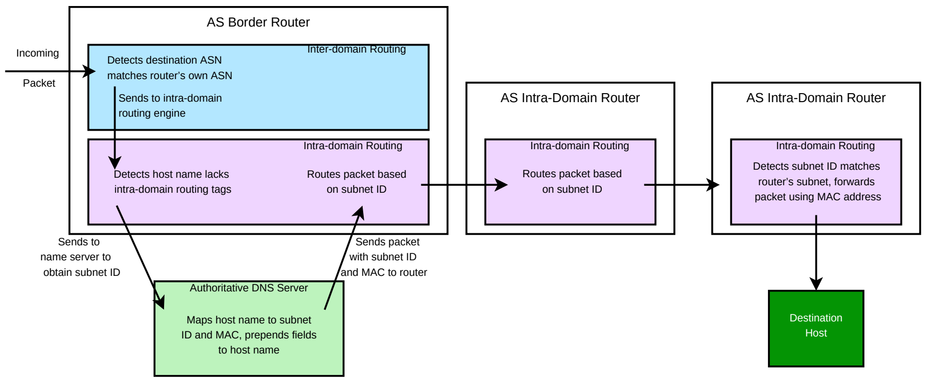
AS Border Router
Incoming
Packet
Inter-domain Routing
Detects destination ASN matches router's own ASN
Sends to intra-domain routing engine
Intra-domain Routing
Detects host name lacks intra-domain routing tags
Routes packet based on subnet ID
Sends to name server to obtain subnet ID
Sends packet with subnet ID and MAC to router
Authoritative DNS Server
Maps host name to subnet ID and MAC, prepends fields to host name
AS Intra-Domain Router
Intra-domain Routing
Routes packet based on subnet ID
AS Intra-Domain Router
Intra-domain Routing
Detects subnet ID matches router's subnet, forwards packet using MAC address
Destination Host

Figure 3: Intra-domain routing process

In Figure 3, we show how routers within a destination AS would direct a packet to the intended host. When the packet arrives, the router would recognize that the destination AS matched the router's own AS. It would then direct the packet to the intra-domain forwarding engine. The intra-domain engine would examine the host name field looking for a reserved value indicating the subnet ID and MAC address have been embedded. On the first packet for a connection, this information would not be present. The router would then send the packet to the network's authoritative DNS server. That DNS server would then perform a lookup and map the host name to its respective ID and MAC address[3]. Once mapped, the name server would alter the packet's destination host name and embed a reserved value indicating the subnet ID and MAC address were set. The name server would then insert these two values, followed by the original destination host name. At that point, the name server sends the packet back to the intra-domain router. The intra-domain router can then look at the destination host name, recognize the subnet ID, and send the packet to the appropriate next hop. Subsequent intra-domain routers forward the packet on the subnet ID until it reaches a router in the subnet. That final intra-domain router forwards the packet to the host by crafting a packet with a link-layer header containing the MAC address encoded in the address. At this point, the destination host can receive the packet, extract its own host name, and confirm it is the destination. The recipient can simply swap the source and destination host name in replies it sends. The original sender can then use the source host name, which will include the subnet ID and MAC address, in subsequent packets, allowing the host to bypass the destination network's name server.

When sending a packet, an end-host can begin the process by encoding their subnet and MAC address with the host name, allowing the response to bypass their own name server. Accordingly, the mapping infrastructure is only needed for the first packet in the connection, much like modern DNS. However, unlike modern DNS, the lookups are divided: the host name to ASN mapping takes place in a regular DNS lookup and is shared across the Internet. However, the host name to subnet ID and MAC mapping is confined to the destination AS. This separation of mapping infrastructure allows DNS resolvers to perform

[3]If no entry exists, the name server may simply discard the packet and optionally send an ICMP message indicating the error.

extensive, long-term caching for host name to AS mappings, which are fairly static. At the same time, host name to subnet ID and MAC mapping, which may change often, is not cached. This allows an authoritative name server to load balance a host name across systems.

These changes to intra-domain packet routing and forwarding affect the routers and the protocols they use to communicate. These engines must be replaced with data structure accommodating the session ID and have logic to divert packets to the name server and to forward using encoded MAC addresses. The routing protocols must be altered to exchange session IDs; however, these changes are not substantial, making replacement routing protocols straightforward. The current protocol could be retained but the meaning of the exchanged information can be reinterpreted. For example, in an AS that choses to use a 16 bit subnet ID, the routers could exchange 16 bit prefixes with the network portion representing the subnet ID, yet recast the leading 16 bits of the prefix to a subnet ID when writing them to the forwarding tables.

3.5. Intra-domain Protocols

The changes described in this work require modifications in intra-domain protocols, such as the dynamic host control protocol (DHCP) and the address resolution protocol (ARP). However, the changes to these protocols are straightforward.

In DHCP, the DHCP server must communicate the network information to a client. The server must provide at least one DNS host name, a subnet ID, and the encompassing ASN to the client. The DHCP server must communicate this mapping information, along with the client's MAC, to the authoritative DNS server the first time it assigns the mapping and each time it changes. Unlike IP addresses, which are often timeshared, DNS host names are less likely to require timesharing and can be retained for a longer period of time. Alternatively, the DHCP server can provide the authoritative DNS server with a TTL value for the mapping record.

Like DHCP, changes to the ARP protocol are straightforward. Rather than map a link layer address to an IP address, the modified ARP will map to a host name and vice versa. However, dynamics involving ARP also change. In IP, hosts could use the subnet information to determine whether a host was in the same subnet, allowing it to know when to use ARP. With host names, hosts on the same subnet may have completely distinct host names. As a best effort, hosts can ARP for other hosts that share the same domain name. Otherwise, the host would issue a DNS request for the host name. When providing results to a host inside the AS, the local resolver can additionally provide the subnet ID and MAC address in addition to the AS information. When the client receives this response, the host can confirm the destination is in the same subnet and use the supplied MAC to reach the destination. Accordingly, unless hosts use DNS names with a hierarchical structure, the usage of ARP may substantially decrease. However, this will not introduce increased reliance upon the local resolver, since this resolver is already required to map host names to IP addresses in the modern Internet.

3.6. The New DNS

The DNS today contains at least 42 different record types [19]. This includes records to find mail servers and authoritative DNS servers for the domain, records to map host names to IP addresses, and records to map IP addresses to host names (reverse DNS mapping). Each of these records have a time to live (TTL) associated with it, allowing the client resolvers to determine how long to cache the record. The DNS functionality required under our architecture is simpler since only one type of record is required to map host names into routing locators. In previous work, we found that domains are largely co-located [18], frequently resulting in just a single routing locator for each domain, or in rare cases, a small number of locators. Thus, the DNS response for `www.example.com` may simply be "`example.com X 24`", where `X` is the ASN locator for the domain `example.com` and 24 is the TTL for the record. This simplicity has two outcomes. First, name servers will have to maintain very few records to represent all of their hosts. Second, since one response covers all the hosts in that domain or sub-domain, client resolvers only have to get the locator once for all the hosts in that domain until the TTL expires. This thwarts statistical and related data attacks to poison the DNS cache, since only a single record can be obtained. Further, since domains tend not to change provider networks frequently, the TTL times for these records may be longer than the TTLs used in the more fine-grain DNS records of today. This significantly reduces the number of queries the DNS server must field from clients.

Other DNS record types simply leverage the DNS as a distributed database, but are not actually a key component of the system. However, the proposed architecture can use reserved host names to support these records as well. The `MX` record is used to find the mail servers for a domain. In our scheme, we can reserve the host "mail" in each domain to serve as an alias to the domain's mail server, if it exists. For example, "mail.example.com" would map to a number of other mail server names (e.g. mail-1.example.com and mail-2.example.com). When the packet crosses the authoritative DNS server at the destination, any of these delegates could be selected and the subnet ID and MAC address supplied in the destination address. Other record types, such as `RP`, `SRV`, and `TXT` records, as well as other less common DNS records, would use host delegations, much like MX records, each with their own reserved host name.

4. Examining ASNs as Locators

Currently, there is an order of magnitude fewer ASNs in the Internet than there are IP prefixes. These ASNs have had with linear growth in recent years [20]. While this bodes well for smaller forwarding tables at the core routers, we must examine the issue of ASN growth carefully: if ASNs grow tremendously and overtake the growth in the number of prefixes, all the benefits would be lost. Here, we examine the issue of ASN growth, approaches to maintain modern traffic engineering goals, the size of resulting forwarding tables, and packet forwarding speeds.

4.1. Growth in ASNs

As the Internet grows and new administrative domains are formed, some growth in the number of ASNs is inevitable. In Figure 4, we show historical ASN growth and a linear fit for these results. While some growth may be required, other factors also affect ASN growth and their impact needs to be carefully examined. For example, work by Huston indicates that ASN growth is fueled by the growth in multi-homing at edge networks [21, 22]. However, locator-identifier split architectures using ASNs as locators will not exhibit such growth.

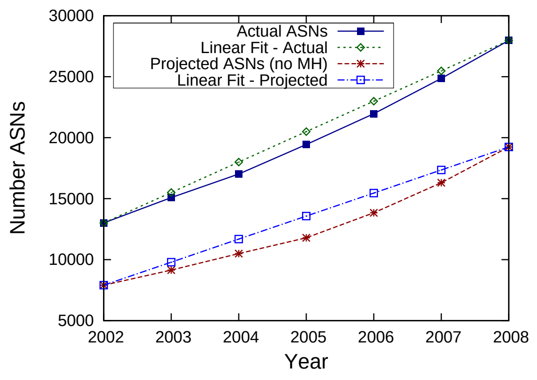

Figure 4: Historical ASN growth and project growth without multi-homing

In our scheme, organizations need not acquire ASNs simply in order to multi-home. Instead, the organization can simply rank each of its providers, indicating the primary provider, secondary provider, and so on. The organization would then simply add each of its providers and their ranks to a mapping database entry for the address range. Upon receiving a packet destined to that organization, the encapsulating router would consult the mapping database and select the provider with the highest priority. If that provider becomes unreachable, the provider with the next highest priority will be selected automatically. Accordingly, an organization can obtain the benefits of multi-homing, yet not have to participate in BGP, nor acquire an ASN, nor inflate inter-domain routing tables.

We now estimate how many of the ASNs in the current Internet exist primarily for multi-homing. Such stub networks would not require an ASN in our scheme. While our approach is necessarily conservative, *we find that almost a third of the ASNs in the current Internet exist solely for the purpose of multi-homing. These ASNs are unnecessary in our architecture and can be eliminated, aiding scalability.*

Methodology: To estimate the number of multi-homed ASes, we identify ASes composed solely of multi-homed prefixes. We use the approach by Bu *et al.* [14] to determine a multi-homed prefix. In this approach, a prefix is considered multi-homed if and only if that prefix is a subset of a prefix from a neighboring AS. Accordingly, we consider an AS to be fully multi-homed if all of the prefixes it originates are sub-prefixes of neighboring ASes. This approach does not help in identifying ASes that use provider-independent prefixes for multi-homing and hence causes us to under-estimate the number of multi-homed ASes.

We use two types of data from the Route Views Project [3] in order to perform this analysis. The first is a BGP RIB from November 15, 2008. From this RIB, we determined which AS originates each prefix. Next, we obtain all the BGP updates during the entire month of November for each of the 42 Route Views vantage points. We examine the AS path in each routing update to determine the peers for each AS[4]. For each prefix, `X`, in the RIB, we determine which ASes, if any, have a super-prefix, `Y`, that encompasses the prefix. If the ASes originating prefixes `X` and `Y` are directly connected, we consider prefix `X` to exist for the purpose of multi-homing. If a stub AS is composed solely of multi-homed prefixes, that AS is considered to exist primarily for multi-homing.

Results: *We find that 8,732 (31.2%) of the* $27,974$ *ASNs in the Internet primarily exist for multi-homing. This estimate is a lower bound because we are unable to infer multi-homed ASes that do not use provider-dependent addressing.* These results suggest that our scheme would require only $19,242$ ASNs.

[4]Some links may be missed if no associated updates were issued during the month snapshot. This would cause an underestimation of multi-homing.

With the widespread usage of provider-independent prefixes for multi-homing, likely fueled by address fragmentation, this is likely a significant underestimate of the amount of ASNs that could be reclaimed. Further, since modern ASN growth is largely expected to be fueled by multi-homing and since such growth would not affect ASNs in our scheme, ASN growth in our approach may dramatically decrease. In Figure 4, we have shown the projected growth with multi-homing eliminated. While largely a vertical displacement from the current ASNs, we note that the projected ASN growth without multi-homing is slower than modern growth ($1,890$ ASNs per year vs. $2,494$) even with these conservative estimates.

4.2. Addressing Loss of Precision in ASNs

Modern ASNs are typically associated with several IP prefixes. This is caused by several factors, as outlined in Section 2, which include *address fragmentation*, *failures to aggregate*, *load balancing*, and for other traffic engineering. While the first two factors may not be inherently valuable, load balancing and traffic engineering are powerful tools that must be accommodated in these architectures. We now examine how each of these factors impact forwarding table size.

While the routers' failure to aggregate prefixes that are aggregatable hinders modern routing, it does not impact inter-domain routing under locator-identifier split proposals because these prefixes are mapped to the same aggregated locators by the encapsulating router. Similarly, address fragmentation is also resolved in the mapping table: while each prefix would require separate mapping table entry, they would result in the same locator being used, masking the growth from inter-domain routers. Accordingly, neither failures to aggregate nor address fragmentation would affect inter-domain forwarding table size or ASN growth. Further, this aggregation would not result in a loss in routing flexibility.

Load balancing at the originating AS, like multi-homing, can leverage the mapping database to avoid causing growth in the inter-domain forwarding table. As in the case of multi-homing, an organization may associate multiple providers with its address range in a mapping database entry. However, unlike multi-homing, which ranks the providers to indicate the primary provider, load balancing would use the same rank for multiple providers. When an encapsulating routing processes a packet destined to such a load-balanced address range, it consult the mapping database, independently and randomly selects one of the associated locators, and uses that locator in all subsequent packets to that address range. Alternatively, the mapping router could partition the host name space for the destination and route each partition via separate routes. These approaches facilitate traffic engineering while avoiding route fluttering and growth in the inter-domain forwarding tables.

Once mapped, subsequent inter-domain routers have only an AS number to direct packets. Previously, these routers could divide traffic to a destination AS by treating each prefix associated with that AS independently. Accordingly, load balancing or special traffic engineering protocols, such as MPLS [23], could exert fine-grain control. Unfortunately, with AS numbers as destinations, routers could only exert course-grain control. In this scheme, these routers have only three options: 1) accept the course-grain limitations of ASNs, 2) probabilistically divide traffic to an AS and induce route fluttering, or 3) leverage the name layer to help make decisions. In the third case, routers could select an arbitrary set of bits in the destination host name and use these to divide traffic. While such operations would be quick and avoid ASN growth, it would add slight complication to forwarding operations.

An alternative to addressing the course-grain traffic engineering problem would be to split each existing AS into multiple ASes, increasing the number of forwarding table entries. Routers would then be able to use the ASN for forwarding, yet retain fine-grain control. Accordingly, we must estimate such growth when calculating the forwarding table size.

Today, a forwarding table entry at the core routers is comprised of IP prefix and the associated next hop information. Under our architecture, it will be the ASN and its associated next hop information. A simple way to estimate forwarding table sizes under our architecture would be to count the number of ASNs advertised in the Internet and subtract the ASNs that exist solely for multi-homing purposes. However, doing so would fail to account for growth caused by *traffic engineering*, in which ASes use multiple distinct paths to route their traffic. While simple load balancing can be accomplished in our scheme without causing growth, other traffic engineering may cause growth in forwarding table sizes and must be examined.

We now estimate the number of forwarding table entries under our scheme. *We find that even without optimizing modern routing for our architecture, forwarding tables under our scheme would require 35.1% of the forwarding table entries in modern routers, even after accounting for traffic engineering.*

Methodology: To estimate the number of forwarding table entries in the presence of traffic engineering, we examined all update messages received by each of the 42 vantage points in the Route Views Project during the month of November, 2008. For each update, we recorded the originating AS and the path used to reach the advertised prefix. If a stored prefix was updated, we deleted the old entry and stored the new entry. To exclude simple load balancing and solely multi-homed stub ASes, which do not increase ASNs in our scheme, we applied rewriting rules to the route updates. Specifically, load balanced IP prefixes were rewritten as their aggregated prefix and solely multi-homed ASes were replaced in the AS path by the appropriate provider AS for the prefix.

During a routing change, some updates to the prefixes for an AS may not be atomic. Accordingly, some prefixes may be updated to the new path while others still reflect the old path, leading to a temporary increase in path diversity for an AS. We regard this transient state as an artifact of current routing practice and not as a traffic engineering goal. To exclude this inflated diversity, we performed periodic snapshots of the routing table after a brief period of inactivity. This analysis allowed us to estimate the number of unique paths used to reach each AS originating a prefix. Since each path is potentially an indication of traffic engineering, it allowed us to estimate an upper bound on the number of entries per ASN.

Results: Our data had information about 30,672 ASes and 288,685 prefixes. We found that 27% of the ASes had a median of one unique route, indicating that the AS path was identical for each prefix originated by that AS. These ASes could be summarized by a single entry in the forwarding table. An additional 25% of ASes had a median of two unique routes, indicating that an extra AS entry would be required to exactly duplicate modern traffic engineering goals. In total, 76% of ASes would require 4 or fewer entries and 94% of ASes would require 10 or fewer entries. *Accounting for each extra entry due to traffic engineering after excluding load-balancing and solely multi-homed ASNs yielded a total of 101,310 entries, which was approximately 35.1% of the 288,685 prefixes in the BGP forwarding tables at the time. This indicates that in spite of traffic engineering, the forwarding tables at the core routers under our scheme will have about one third the number of entries in the worst case.*

4.3. Forwarding Table Lookup Performance

Today, forwarding table entries consist of IP prefixes of variable lengths and routers perform a *longest prefix match* to determine the interface for a packet. Under our scheme, packet forwarding will occur on *fixed length* ASNs. Now, we examine the impact of this factor on packet forwarding speeds. *We find that ASN-based packet forwarding makes lookup and update operations an order of magnitude faster while requiring less than one-third of the memory requirements of IPv4 forwarding in software routers.*

Methodology: Modern software routers use the trie data structure to perform longest prefix matching on IP prefixes [24], which are variable in length. A trie must perform $O(log(n))$ memory references, where n is the number of bits in an IP address. These software routers are an important area of research where high performance is required [25, 26]. However, hardware routers often use Ternary Content Addressable Memory (TCAM) to store their forwarding tables to store entries. With these data structure, each lookup can be performed in parallel and still yield a constant lookup ($O(1)$). Unfortunately, TCAMs are 1) expensive to produce, 2) limited in capacity, and 3) draw significant electricity, which is becoming an acute problem [27]. ASN-based routing differs from both approaches because ASNs are *fixed length*. Thus, exactly one match has to be found. To exploit the fixed-length nature of ASNs, we explore a hash table lookup method. This approach requires a single memory reference in the absence of collisions, yielding a performance of $O(1)$. Accordingly, the performance of ASN-based packet forwarding would be similar to simple hash table performance and could be efficiently implemented with low SRAM density, yielding fast lookups with lower cost and energy consumption. Because we lack access to tools for hardware testing, we focus on comparing the software implementations of longest prefix matching and ASN-based forwarding approaches.

Until 2006, ASNs were two bytes in size, allowing for direct indexing into the forwarding table: the destination ASN could simply be used as an index into an array of 2^{16} entries. If each forwarding table entry required only four bytes of next-hop information, this would require a mere 256KBytes, minimal computation, and a single memory reference, yielding almost optimal performance. Recently, 4-byte ASNs have become available [6]. Since this larger address space was designed for future growth, it is essential to include the 4-byte representations in our performance analysis. *Accordingly, we assume 4-byte ASNs subsequently.*

To compare the performance of routing in our approach with current routing, we use software implementations of lookup algorithms. In practice hardware implementations are used to accelerate forwarding lookups because hardware can yield faster memory accesses, can facilitate parallelism, and accelerate operations such as hashing. While we are unable to implement these approaches in hardware, the software implementations serve as a lower-bound on performance and can show the potential benefits of a new algorithm.

To evaluate the hashing approach, we use a hash table implementation, the `unordered_map` data structure from the TR1 C++ library, to store and access entries. For a baseline comparison, we use a software implementation of a Tree Bitmap trie for IPv4, which is described in detail in our prior work [4].

To populate the ASN-based hash table, we store an entry for each of the ASNs required for the forwarding table described in Section 4.1. This requires 101,310 entries. For the IPv4 baseline, we load each of the prefixes found in a November 15, 2008 routing table for a Route Views router, which is from the same time as the ASN analysis. This routing table had 288,685 IP prefixes.

All performance tests were done on a machine with a Pentium IV 3.2 GHz processor with 2GBytes RAM. To measure the timings, we use the `RDTSC` instruction, which can be used to measure the elapsed cycle count, yielding nanosecond timing resolution.

Results: In Table 1, we show the results of our ASN and IPv4 models. In the second column of Table 1, we show the results of the ASN approach. We note that this results in very low memory usage. The performance of this scheme is excellent, requiring only 155ns on average to perform a lookup or an update operation. In the third column, we show the results for the IPv4 baseline. We can see that the IPv4 routing table requires over three times as much memory, about 9.73 MBytes, and takes longer to perform a lookup, averaging about $1,129$ ns.

Forwarding Approach:	ASNs	IPv4
Number of entries:	101,310	288,685
Storage required (MBytes):	2.90	9.73
Lookup Times (ns):		
Average:	155	1,129
Standard Dev.:	40	253
Minimum:	133	543
Update Times (ns):		
Average:	157	4,018
Standard Dev.:	96	1,283
Minimum:	134	2,528

Table 1: Performance of ASN-based and IPv4 forwarding.

From this analysis, we confirm that ASN-based lookups have excellent performance and with hardware optimizations may greatly accelerate the packet lookup process, access control list (ACL) processing, and additional data-plane operations, expediting packet forwarding.

5. Examining Host Names as Identifiers

Using host names as end-host identifiers impacts DNS latency only in positive ways. This is because in our scheme, locators exist at domain or sub-domain granularity and requests need not be sent to remote authoritative DNS servers for each individual record. Beyond this, the usual DNS caching practices today continue to be effective and may be able to use longer TTLs since domains tend not to change as frequently as individual records. Also, since the inter-domain routing component of our architecture leverages the well-researched NIMROD, LISP, eFIT, and ENCAPS-like proposals, we assume that inter-domain routing is scalable. We now examine two other issues, namely, the impact of increased layer three packet header size and feasibility of conducting intra-domain routing on host names.

5.1. Packet Header Growth

Methodology: Using host names as addresses may result in higher packet header overhead. In order to calculate this overhead, we need to know the length of an average domain name. To determine this, we collected *zone files* from the `.com` and `.net` generic top-level domains (gTLDs) on June 8, 2007. These zone files contain each second level domain registered under these gTLDs on that day. Ideally, we would have liked to have the zone files from the rest of the gTLDs and all the country-code TLDs (ccTLDs). However, getting access to that data was not possible. To compensate, we collected data from the DMOZ Open Directory Project [28]. The project contains user submitted links and is the largest and most comprehensive directory of the Web. Our input data, collected on October 28, 2006, has 9,633,835 unique URLs and 2,711,181 unique second and third-level domain names. We then combined the DMOZ data and the zone file data, eliminating any duplicate domains. Combined, the data set contained 79,088,314 unique domains. This represents a significant percentage of the 128 million total domains in the Internet in June 2007. Using this data, we determine the distribution of the number of characters in a domain name.

Results: While the maximum number of characters in a host name could be 255, individual labels separated by the '.' character could be up to 63 characters. In our data, the maximum length of any domain was 67 characters. The distribution was roughly normal, with a median of 15 characters long, shown in Figure 5. If hosts within domain names follow a similar pattern, the host name would be a median of about 30 characters. Further, due to restrictions in the DNS character set (only case-insensitive letters, numbers, dashes, and periods are valid), only 6 bits are required to encode each character, resulting in a 23 byte address. Along with an extra byte to encode the name's length, host addresses would be 24 bytes, which is 50% larger than an IPv6 address. If the subnet ID and MAC were encoded in the host name as well, this would add around 7 bytes, yielding a 31 byte name. While larger, these addresses can be easily fit within modern packet size limitations.

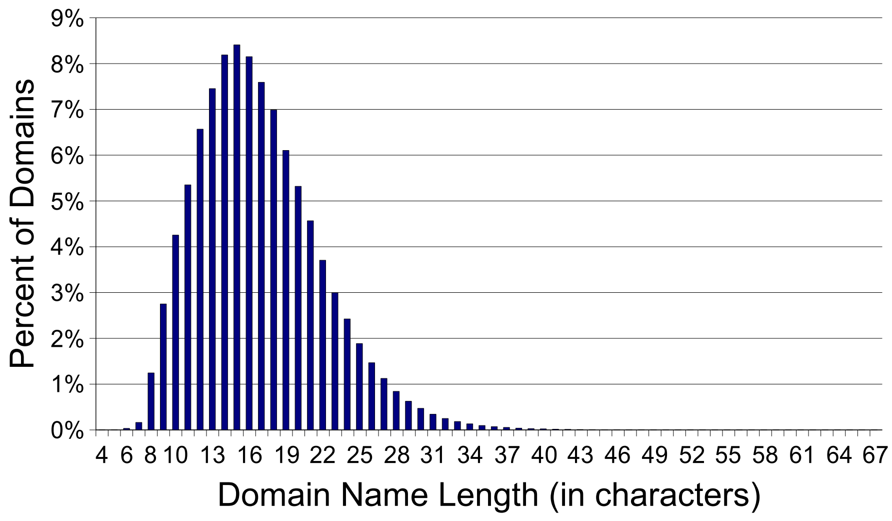

Figure 5: Length of domain names in the Internet

5.2. Intra-domain Routing Scalability

In our architecture, once traffic reaches the destination network, it must be forwarded based on its host name. Since destination networks may be large, we must examine the issue of intra-domain scalability closely. Packet forwarding would use three identifiers: 1) the host name, which a name server must be able to quickly map to a subnet ID and MAC, 2) a subnet ID which must be quickly routed, and 3) a MAC address for transmitting the packet at the last router. Fortunately, these overheads are easily managed by a network.

Modern DNS servers must currently map a large number of host names to lower level identifiers. In our approach, this step is even less work: the servers would only store one record type, requiring only about 8 bytes of results for each record (2 bytes for the subnet ID and 6 bytes for the MAC address). However, these DNS servers would be provisioned on an AS-wide basis rather than tied to individual domains. Large ASes with a large number of entries may need multiple machines to store these records. However, since these entries can be arbitrarily divided alphabetically, these lookups could easily be load-balanced across multiple authoritative servers.

Routing on subnet IDs and MACs is straight-forward for routers. The subnet IDs, which are likely to be 16 bits or smaller for most ASes. In such cases, direct array indexing can be used, which requires only 384 KB of memory for 65,536 entries and yields almost optimal performance on routers. For MAC address transmission, routers must simply copy the MAC address to the link layer, eliminating the need for any mapping overheads.

6. Discussion of Issues

6.1. Integrating ASNs as Locators

Here, we discuss aspects of integrating ASNs as locators in the locator-identifier split proposals. In some protocols, such as ENCAPS and LISP, the IP packet header is reused for inter-domain routing in order to facilitate deployment. In this case, the locators would be 4 byte values and be placed in the IP header as source and destination addresses. Since both architectures require packets to be encapsulated before they reach the inter-domain routers, ASNs can be used directly as the locators without being confused with the IPv4 addresses used for end-hosts. Since the ASN fits into the IPv4 address space, routers using modern IP forwarding can operate without changes. Protocols that do not use IPv4 encapsulation have more latitude on how to incorporate the ASN in their locators.

When forwarding packets, the routers simply consult the locator layer and use the destination ASN address for packet forwarding. To make such decisions, the routers must be able to map ASNs to out-going interfaces. BGP performs such mapping from prefixes to out-going interfaces. In LISP and eFIT, the forwarding table simply maps ASNs to out-going interfaces. When a router receives a packet destined to a host within its own ASN, it must remove the locator layer and use the embedded network layer information to forward the packet to the end-host. In our architecture, the hosts perform the mappings. However, we justify this design decision by comparing push and pull-based mappings for network-wide resolvers and showing the scalability benefits.

6.2. Pull vs. Push-based Databases

The database that maps end-host identifiers to host names is an important consideration in our architecture and in locator-identifier split proposals. Some approaches, such as LISP and CRIO [29], use a push-based mapping database. These systems require routers to store mappings and perform them on any packet that arrives. Other approaches, such as the DNS, ENCAPS, and our architecture, use a pull-based mapping and request entries only when needed. A pull-based approach can allow hosts to perform the mappings but does introduce extra delay during connection establishment such as the DNS overheads associated with the modern Internet. Given the importance of this mapping, we examine whether our scheme would be viable with a push-based architecture in which routers perform the mappings. We find that a pull-based database is more scalable.

6.2.1. Push-based Databases

For small mapping databases, a push-based architecture offers greater performance, since all entries are locally available. Unfortunately, as the mapping database grows, the growth increases at each encapsulating router. The work by Krioukov *et al.* [7] shows that such an approach for a mapping database cannot scale long-term, threatening the entire scheme. Here, we look at the overheads that would be incurred to support a push-based database.

Upon startup, all routers would have to learn about the domain name to ASN mappings of existing domains. This could be a large amount of information given the number of domain names in the Internet today. However, this information is not required to be propagated often. The primary source of overhead comes from the addition and deletion of domain names. (The actual domain name to ASN mapping of a particular entry rarely changes.) These changes will have to be propagated in a timely manner, though some delay is acceptable. We now estimate the control plane overheads of these changes.

Data Used: Toward this goal, we take daily snapshots of the `.com` and `.net` zone files from May 17, 2007 to May 24, 2007. These files contain the domain names in these TLDs. We then compare each day with the previous day and determine the number of changes (addition or deletion) between them. This yields 7 snapshots of daily changes, shown in Table 2. Given that the `.com` TLD is the biggest and the busiest [30], we hope to have captured the hardest case.

Analysis: The first key observation from Table 2 is that millions of domains get added or deleted each day in the `.com` and `.net` TLDs. Next, we estimate how many new control plane packets will be needed to propagate these updates. For that, we need to know the length of a domain name. From the analysis in Section 5.1, we note the median character length of 15 characters. Using this information, and the maximum Ethernet packet size of 1500 bytes, we estimate the number of new update packets changes to domain names will require. Table 2 shows the the number of update packets per day and per minute.

To put these update numbers in perspective, we compare them with the BGP update rate. To do so, we examine the snapshots from the Route Views Project [3] from February 15 to March 19, 2007. From this, we took the total number of BGP updates per day for each of 46 sites and divided them evenly to find the number of updates per second. Averaged across the sites, these results showed a daily average of 51.3 updates per minute, with a standard deviation

Snapshot	`.com`	`.net`	Total	Required Updates	
				per Day	per Minute
1	3,849,085	384,726	4,233,811	32,320	22.44
2	2,432,957	250,500	2,683,457	20,485	14.23
3	2,149,471	211,933	2,361,404	18,026	12.52
4	1,078,509	87,914	1,166,423	8,904	6.18
5	3,877,005	351,652	4,228,657	32,280	22.42
6	4,227,177	475,554	4,702,731	35,899	24.93
7	1,650,616	155,533	1,806,149	13,788	9.58

Table 2: Total changes in a day to the `.com` and `.net` zone files

of 24.21. The average maximum rate was 143.65 updates per minute. In fact, the average minimum rate was 31.03 updates per minute, which is greater than the maximum rate in a push-based scheme.

From these results, we find that a push-based database could be supported for today's Internet. However, the overheads incurred in maintaining these records is on the same order as BGP updates, which are considered to occur at an alarming rate. With the scalability concerns of such a database, we question whether such an approach is viable in the long-term.

6.2.2. Pull-based Database

Pull-based mapping architectures have been successful on the Internet. The DNS has demonstrated that pull-based architectures can easily scale to billions of entries on modern hardware. By using a pull-based database for mapping identifiers to locators, routers can scalably provide an encapsulation mapping. Further, work on DNS has found that caching individual DNS records for popular destinations at edge networks has been quite effective in boosting resolution performance [31]. In this section, we analyze whether such caching is also effective for domain to ASN mappings.

Data Used: To judge the effectiveness of caching domain name to ASN mappings, we obtained a log of all the DNS requests for a week starting June 13th from clients within our department. These logs indicate the time the request that was made and the hostname being resolved. Table 3 shows an overview of the data. We note that the total number of unique domains queried by hosts from within our department in an entire week is 4 orders of magnitude less than the total number of domains on the Internet, which greatly supports the caching approach.

Start Date	June 13, 2007
End Date	June 19, 2007
Number of Queries	2,991,793
Number of Domains Queried	29,947
Total Number of Domains	128 million

Table 3: DNS Query Information

Analysis: In Figure 6, we plot a cumulative distribution function (CDF) of the percentage of DNS queries covered by the given percentage of DNS domains. We note that this graph only shows the most popular 20% of domains, yet reaches a coverage of 97% of the requests. In fact, caching only the $1,200$ most popular domains (4% of the unique domains requested) would yield a theoretical cache hit rate of 88.89%. These results are similar to the work in [31], which found that the most popular 20% of host names accounted for about 80% of the queries

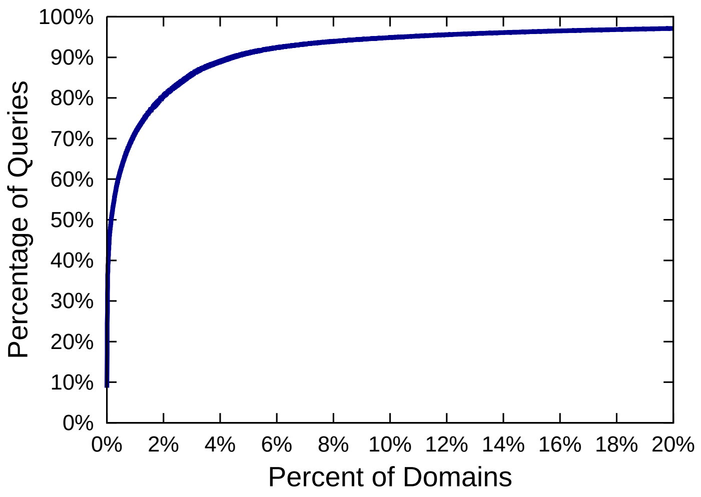

Figure 6: CDF of the percentage of DNS queries for top percentages of domains

While Figure 6 supports caching, it does not account for any churn in the cache resulting from client usage, which could make it hard to leverage the benefits of caching. To examine cache churn, we used a cache of $1,200$ entries. (We realize that the limits we impose on the cache size are artificial since an average edge router should be able to cache all the unique domains accessed in our case. However, we use these limits to test cache churn anyway.) We found that a simple cache using a least recently used (LRU) eviction policy achieves a cache hit rate of 86.02% on our data. We conclude that caching popular domain to ASN mappings appears to be a fruitful approach.

With the effectiveness of caching and lower mapping update overheads, we recommend a pull-based database in resolvers in order to ensure scalability, should hosts require such support.

6.3. Partial Deployment and Transition Plan

With all the networking infrastructure in place, any new Internet proposal must be effective even when partially deployed. However, substantial changes to IP routing have been successful, even though they are not as widely recognized. Multiprotocol Label Switching (MPLS) [23] is commonly used in transit ASes for traffic engineering. This protocol encapsulates IP traffic within an AS and decapsulates the packets before leaving the AS. MPLS can be used both as a tool and as a model for deployment of our architecture. While currently limited to intra-domain traffic, AS numbers can be used as labels in MPLS and enable inter-domain MPLS deployments, allowing course-grain traffic engineering without requiring packet encapsulation and decapsulation at each AS boundary. The first AS in a string of MPLS-aware networks would effectively be performing the encapsulation step. In such an architecture, the MPLS header would represent half of the ASN layer, providing the information for the destination, but not the source. Before reaching the first legacy AS or after entering the destination AS, the MPLS header would be stripped as it is today, with the packet routed on the destination network layer address.

Leveraging inter-domain MPLS provides a transition plan for routing on ASNs, allowing packets to cross multiple networks without consulting the network layer. However, these routers should also implement a revision to the protocol to allow a source ASN to be specified as well to later support end-to-end deployments and avoid the mapping overheads at routers for replies. With inter-domain MPLS and transit packet routing on ASNs, the network layer header becomes decoupled from routing, allowing incremental partial deployment between edge networks. For example, two networks could use IPv6 or name-based routing headers to communicate across a network-layer agnostic core as long as both edge networks support the name layer protocols. This core support is essential to innovations in host identification.

While the core routing infrastructure is being updated, hosts can begin supporting the ASN header and the name-based routing headers. These hosts can be provisioned with this support long before the protocol is used. IPv6, for example, has had support in many modern operating systems for years, despite its lack of widespread deployment. At the same time, name servers can be provisioned with the extra DNS records required for configuring hosts, namely the ASN records for domains. This information can be provided in response to queries, but would not yet be used by hosts.

Once ASN header support is in place, edge networks can begin supporting the name-based routing headers and subnet identifier mapping. These networks can detect whether a given destination network 1) supports the name-based routing header and 2) whether all transit networks on the path support the ASN layer using BGP. Each originating AS can place a transitive, optional, globally-reserved BGP Community value in their routing advertisements to indicate the AS supports the name-based routing header. It would place an additional non-transitive BGP Community value in the advertisement indicating the full path to the AS is compliant. Any router receiving these Community values would forward the information indicating the originating AS is compliant. Any router supporting ASN headers would place the BGP Community in the advertisements they forward only if the incoming advertisement already had the Community present. However, legacy routers would not add this Community value in the announcements it forwards. Accordingly, if a route advertisement arrives with both Community values, it indicates the originating AS supports the ASN header and that all routers on the path to that AS do as well.

Once an AS has confirmed that a destination network and path support the ASN header, it can allow its hosts to begin using the ASN and the name-based routing headers. When a host performs a DNS request, the ISP's DNS resolvers act as a proxy for the request, caching results as needed. With help from the inter-domain routers, these resolvers can determine whether the indicated destination AS can be reached with the ASN and name-based routing layers. If so, the resolver can add DNS records indicating this fact. The host can then decide to either use legacy IP traffic or use the ASN and name-based routing layers. These hosts could be configured with a default value to prioritize the ASN/name-based routing approach after deployment is widespread.

Once most hosts have enabled support, organizations can begin removing legacy IP addresses for individual hosts and begin leveraging techniques such as NAT to provide backward compatibility to the remaining hosts using IP traffic. Eventually, the IP traffic will decrease to the point where organizations can discontinue IP usage, forcing the remaining hosts to upgrade.

While the transition from the modern IP network to name-based routing will be slow, it can be done gracefully allowing networks to independently opt in while facilitating pair-wise edge network deployment using inter-domain MPLS-based ASN routing layer tunnels.

6.4. Host Mobility

Host mobility is becoming increasingly important with smaller devices that travel from network to network. Approaches to separate location and identity can naturally support mobility. The Mobile IP approach [32] is designed for host mobility in IP networks and leverages tunneling. However, when NIMROD, LISP, ENCAPS, or eFIT are used, the destination router can update the locators in the inter-domain routing layer in packets destined to mobile hosts and forward them to the host's visiting network. In replies, the mobile host can include the visiting network locator, allowing the other host to learn its new location, and avoiding triangular routing. While our approach is compatible with host mobility, the underlying inter-domain routing protocol is the appropriate place to incorporate mobility functionality and we rely upon these mechanisms to support it.

7. Related Work

In this work, we leverage ASNs for inter-domain routing and packet forwarding and use DNS host names for host identification. These identifiers have been suggested in two prior works, HLP [17] and TRIAD [33, 34], respectively. In HLP, the authors propose a successor to BGP which provides greater routing scalability and uses announcements at the AS granularity rather than on individual IP prefixes. Our approach could directly leverage the HLP efforts to perform inter-domain routing. However, HLP has a different goal than our work: it seeks to prove the feasibility of a new routing protocol while our approach assumes an existing routing protocol and discusses the changes needed to the protocol to accommodate our new Internet design. While HLP is designed to work with IP traffic and router forwarding tables would still need to map IP addresses to prefixes, the protocol could be modified to incorporate ASN-based forwarding tables as well. To a lesser extent, GIRO [35] leverages AS numbers and combines them with geographical location to aid in shortest physical path routing. While the GIRO scheme focuses on a different problem, our analysis may provide insight on GIRO deployment.

In TRIAD, the authors proposed routing on domain names directly, with a focus on content distribution and caching. The protocol performs name-based routing above the network layer and uses IP addresses as ephermal routing tags to forward packets between TRIAD hosts. Once the packet arrives at a TRIAD host, that machine performs a name-based lookup to determine if it has the content locally, and if not, locates the closest TRIAD node that is likely to have the content. Our approach resembles TRIAD in that we use host names for identifying hosts and we leverage network-layer tunneling to reach the destination. However, the approaches are quite different: TRIAD aims to maintain the existing IP scheme while our approach is designed to transition away from IP addresses entirely. Our approach reduces the state at inter-domain routers and expedites processing at intra-domain routers.

The coupling of locators (IP prefixes) and identifiers (IP addresses) in the current Internet has been widely recognized as major weakness of the current Internet architecture. Accordingly, multiple works have focused on separating locators and identifiers, including eFIT [10], LISP [9], ENCAPS [11], and IS-LAY [36]. We described them in detail in Section 2. In CRIO [29], the authors utilize tunneling between points of presence, which are are far fewer than the number of prefixes, yielding smaller forwarding tables sizes. The end goal of each of these approaches is to reduce the number of entries in the forwarding tables at the core routers.

Changes to end-host addressing has been the focus of several works, including IPv6 [37], GSE [38], IPNL [39], SNF [40], Layered Naming [41], and ROFL [42]. The goals in these works range from simply expanding the address space, as in IPv6 and GSE, or showing that hierarchical addressing is not essential for routing, such as in ROFL. In IPNL, the authors seek to formally integrate NAT into the Internet by using encapsulation and manipulating DNS behavior. In prior work [43], we examined the feasibility of using host names directly for inter-domain packet forwarding. This analysis showed us that simply forwarding packets using host names would not scale across the Internet. However, this work provided valuable background for this work. In HIP [44] and AIP [45], the authors use public key cryptography to create secure host identities. Seamless end host mobility has been a subject of research as well. Works, such as FARA [46] and i3 [47], have focused on end-host mobility and utilize rendezvous mechanisms to facilitate communication.

The compact routing field has evaluated the long-term scalability of many routing approaches. In the work by Krioukov *et al.* [7], the authors note that ASes are a natural choice for locators and that there are an order of magnitude fewer ASes than the number of announced prefixes. The transition to ASNs would immediately reduce forwarding tables by an order of magnitude, which would relieve our current concerns about router forwarding table capacity. However, the authors caution that this could be simply a one-time benefit and indicate that the rate of growth of ASes exceeds that of IP prefixes. While raising this concern, the authors did not examine the causes of this AS growth. However, upon considering the causes of growth, we find that under a split locator and identifier scheme using ASes several growth factors would be eliminated, slowing ASN growth. Further, the Krioukov work also indicates the mapping from identifiers to locators requires a global distributed mapping database, reducing scalability. However, a pull-based database, such as the DNS, can perform these mappings in a scalable manner. Accordingly, we believe split locator and identifier schemes merit consideration.

Other works provide insights on the design of next generation architectures. In [48], the authors propose using resilient overlay networks to increase reliability for end-hosts. In this system, end-hosts join small overlay networks which have diverse network vantage points, generally allowing hosts to reach a destination assuming any physical connection exists to the destination. In [49], the authors survey current architecture design options and implications, with a focus on allowing future evolution of the Internet. In [50], the authors advocate a separation of infrastructure providers from service providers by creating virtual networks and allowing infrastructure to be shared by multiple architectures.

8. Conclusion

IP addresses have been a cornerstone of the Internet for as long as we have known the TCP/IP-based Internet. Both host names and autonomous system numbers (ASNs) were added later: host names were added to provide users with a mnemonic way of addressing machines and ASNs were added to make BGP loop free. This paper explored a new Internet which breaks away from IP addresses and instead embraces names as host identifiers and ASNs as locators. This design decouples routing from addressing, which IP addresses (inadvertently) entangled. The outcome is a faster, expandable, and more scalable Internet. We outlined the key features of our architecture and justified the choices using actual data sets from the Internet. While there is still more work required to test the feasibility and to make the architecture practical, we hope that this paper will continue the discussion of the future of the Internet in the research community.

Acknowledgments

We would like to thank Andrew Kalafut for his analysis of the changes in the zone files. We would like to thank D. Kevin McGrath for providing the analysis of BGP update messages. Finally, we would like to thank Rob Henderson for providing the DNS data used in our cache examination.

References

[1] D. Meyer, L. Zhang, K. Fall, Report from the IAB Workshop on Routing and Addressing, IETF RFC 4984 (Sep. 2007).

[2] R. Hinden, S. Deering, E. Nordmark, IPv6 global unicast address format, IETF RFC 3587 (Aug. 2003).

[3] University of Oregon Advanced Network Technology Center, Route Views project, `http://www.routeviews.org/`.

[4] C. Shue, M. Gupta, Projecting IPv6 forwarding characteristics under Internet-wide deployment, in: ACM SIGCOMM IPv6 Workshop, 2007.

[5] Potaroo, Bgp routing table analysis reports, `http://bgp.potaroo.net/` (2008).

[6] Potaroo, The 32-bit AS number report, `http://www.potaroo.net/tools/asn32/index.html` (2008).

[7] D. Krioukov, K. Claffy, K. Fall, A. Brady, On compact routing for the Internet, in: ACM SIGCOMM Computer Communications Review (CCR), 2007.

[8] I. Castineyra, N. Chiappa, M. Steenstrup, The Nimrod routing architecture, IETF RFC 1992 (Aug. 1996).

[9] D. Farinacci, V. Fuller, D. Oran, D. Meyer, S. Brim, Locator/ID separation protocol (LISP), IETF Internet Draft (Jul. 2008).

[10] D. Massey, L. Wang, B. Zhang, L. Zhang, A scalable routing system design for future Internet, in: ACM SIGCOMM Workshop on IPv6 and the Future of the Internet, 2007.

[11] R. Hinden, New scheme for Internet routing and addressing (ENCAPS) for IPNG, IETF RFC 1955 (Jun. 1996).

[12] L. Zhang, S. Brim, A taxonomy for new routing and addressing architecture designs, IETF Internet Draft draft-rrg-taxonomy-00.txt (Mar. 2008).

[13] P. Mockapetris, Domain names - implementation and specification, IETF RFC 1035 (Nov. 1987).

[14] T. Bu, L. Gao, D. Towsley, On characterizing BGP routing table growth, Computer Networks 45 (1) (2004) 45 – 54.

[15] D. Jen, M. Meisel, D. Massey, L. Wang, B. Zhang, L. Zhang, APT: A practical transit mapping service, IETF Internet Draft (Nov. 2007).

[16] E. Lear, NERD: A not-so-novel EID to RLOC database, IETF Internet Draft (Sep. 2007).

[17] L. Subramanian, M. Caesar, C. Ee, M. Handley, M. Mao, S. Shenker, I. Stoica, HLP: A next generation inter-domain routing protocol, in: ACM SIGCOMM, 2005.

[18] C. Shue, A. Kalafut, M. Gupta, The Web is smaller than it seems, in: ACM SIGCOMM Internet Measurement Conference (IMC), 2007.

[19] A. Kalafut, C. Shue, M. Gupta, Understanding the implications of DNS server provisioning, in: ACM SIGCOMM Internet Measurement Conference (IMC), 2008.

[20] A. Dhamdhere, C. Dovrolis, Ten years in the evolution of the internet ecosystem, in: IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2008, pp. 183–196. doi:http://doi.acm.org/10.1145/1452520.1452543.

[21] G. Huston, Analyzing the Internet's BGP routing table, The Internet Protocol Journal 4 (1).

[22] G. Huston, Exploring Autonomous System numbers, The Internet Protocol Journal 9 (1).

[23] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, IETF RFC 3031 (Jan. 2001).

[24] G. Varghese, Recent research directions, `http://www-cse.ucsd.edu/users/varghese/research.html` (2004).

[25] J. Naous, G. Gibb, S. Bolouki, N. McKeown, NetFPGA: Reusable router architecture for experimental research, in: ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow, 2008.

[26] K. Argyraki, S. A. Baset, B. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedveschi, S. Ratnasamy, Can software routers scale?, in: ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow, 2008.

[27] R. Bolla, R. Bruschi, F. Davoli, A. Ranieri, Energy-aware performance optimization for next-generation green network equipment, in: ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow, 2008.

[28] DMOZ, Open directory project, `http://www.dmoz.org/`.

[29] X. Zhang, P. Francis, J. Wang, K. Yoshida, Scaling global IP routing with the core router-integrated overlay, in: IEEE International Conference on Network Protocols (ICNP), 2006.

[30] VeriSign, Domain name industry brief, http://www.verisign.com/static/042161.pdf (Jun. 2007).

[31] J. Jung, E. Sit, H. Balakrishnan, R. Morris, DNS performance and the effectiveness of caching, in: ACM SIGCOMM Internet Measurement Workshop, 2001.

[32] C. Perkins, IP mobility support for IPv4, IETF RFC 3344 (Aug. 2002).

[33] D. Cheriton, M. Gritter, TRIAD: A new next generation Internet architecture, Tech. rep., Stanford Computer Science (March 2000).

[34] M. Gritter, D. Cheriton, An architecture for content routing support in the Internet, in: USENIX Symposium on Internet Technologies and Systems (USITS), 2001.

[35] R. Oliveira, M. Lad, B. Zhang, L. Zhang, Geographically informed inter-domain routing, in: IEEE International Conference on Network Protocols (ICNP), 2007.

[36] F. Kastenholz, ISLAY: A new routing and addressing architecture, IETF Internet Draft (May 2002).

[37] S. Deering, R. Hinden, Internet protocol, version 6 (IPv6) specification, IETF RFC 2460 (Dec. 1998).

[38] M. O'Dell, GSE - an alternate addressing architecture for IPv6, IETF Internet Draft (Feb. 1997).

[39] P. Francis, R. Gummadi, IPNL: A NAT-extended Internet architecture, in: ACM SIGCOMM, 2002.

[40] A. Jonsson, M. Folke, B. Ahlgren, The split naming/forwarding network architecture, in: Swedish National Computer Networking Workshop, 2003.

[41] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, M. Walfish, A layered naming architecture for the Internet, in: ACM SIGCOMM Computer Communications Review (CCR), 2004.

[42] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, S. Shenker, ROFL: Routing on flat labels, in: ACM SIGCOMM, 2006.

[43] C. Shue, M. Gupta, Packet forwarding: Name-based vs. prefix-based, in: IEEE Global Internet (GI) Symposium, 2007.

[44] R. Moskowitz, P. Nikander, Host identity protocol (HIP) architecture, IETF RFC 4423 (May 2006).

[45] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, S. Shenker, Accountable Internet protocol (AIP), in: ACM SIGCOMM, 2008.

[46] D. Clark, R. Braden, A. Falk, V. Pingali, FARA: Reorganizing the addressing architecture, in: ACM SIGCOMM Computer Communications Review (CCR), 2003.

[47] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, Internet indirection infrastructure, in: ACM SIGCOMM, 2006.

[48] D. Andersen, H. Balakrishnan, M. F. Kaashoek, R. Morris, The case for resilient overlay networks, in: IEEE Workshop on Hot Topics in Operating Systems, 2001.

[49] S. Ratnasamy, S. Shenker, S. McCann, Towards an evolvable Internet architecture, in: ACM SIGCOMM, 2005.

[50] N. Feamster, L. Gao, J. Rexford, How to lease the Internet in your spare time, ACM SIGCOMM Computer Communications Review (CCR) 37 (1) (2007) 61–64. doi:http://doi.acm.org/10.1145/1198255.1198265.