HAIR: Hierarchical Architecture for Internet Routing

Anja Feldmann
T-Labs/TU Berlin
Berlin, Germany
anja@net.t-labs.tu-berlin.de

Luca Cittadini
Università Roma Tre
Rome, Italy
luca.cittadini@gmail.com

Wolfgang Mühlbauer
T-Labs/TU Berlin
Berlin, Germany
wolfgang@net.t-labs.tu-berlin.de

Randy Bush
Internet Initiative Japan
Tokyo, Japan
randy@psg.com

Olaf Maennel
Loughborough University
Loughborough, UK
olaf@maennel.net

ABSTRACT

In the light of recent interest in re-designing the Internet, we introduce HAIR, a routing architecture that tackles the problem of routing table growth, restricts the visibility of routing updates, and inherently supports traffic engineering, mobility, and multipath.

HAIR separates locators from identifiers. The routing and mapping system rely on a hierarchical scheme that leverages the structure of today's Internet. Contrary to proposals such as LISP [7] and shim6 [18], we use a hybrid *edge-based* approach where only some lightweight functionality is added within the network, while the majority of tasks are performed as close to the end hosts as possible.

To evaluate our architecture, we analyze to what extent routing would be simplified if HAIR were deployed in today's Internet. Finally, we demonstrate the feasibility of our approach by presenting a working proof-of-concept implementation.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design, Experimentation

Keywords

Future Internet, Scalable Routing Architecture

1. INTRODUCTION

While the research community is still discussing the fundamental limits to routing scalability [13], routing tables in the default-free zone of today's Internet already contain some 300,000 prefixes and continue to grow super-linearly. This goes along with a steady increase in the rate of changes to the routing tables [10]. Moreover, features such as mobility, security, support for traffic engineering or multi-homing are lacking; IP addresses are a scarce resource, and customers are hindered by provider lock-in.

Given all these problems, suggesting a major overhaul of today's Internet has become a popular research topic. We, in this paper, introduce HAIR, a routing architecture that tackles the problem of routing table growth, restricts the visibility of routing updates, and inherently provides means for traffic engineering and mobility.

Despite the multitude of existing proposals for future Internet architectures, there is wide consensus among scientists to separate the *identifier* from the *locator* functionality (Loc/ID split), e.g., [7, 12, 18, 24]. Currently, both are mangled together within IP addresses. Loc/ID split allows for persistent identifiers and at the same time for aggregation in the locator namespace. There exist multiple proposals, amongst others *host-based* (e.g., shim6 [18]) and *network-based* (e.g., LISP [7]) approaches. The main advantage of host-based solutions is that they can be self-deployed without cooperation of the network operators. Network-based approaches, however, are capable of transparently supporting legacy hosts and of reducing routing table size in the core. We note that these alternatives have different objectives and at least in part complement one another. Therefore, we argue for a hybrid *edge-based* solution that transfers to end hosts tasks such as translating identifiers to locators, while keeping some elements of the network-based approaches to address scalability limits and migration issues.

Another key to the design of both the routing *and* the mapping system of HAIR is to leverage the structure which is inherently present in today's Internet [4, 21]: the existence of a stable "core" formed by large transit providers (*CORE*), and a more dynamic edge, consisting of smaller access or enterprise networks (*INTERMEDIATE*, short *INT*) and *EDGEs*, e.g., Ethernet domains. The goal is to prevent both routing updates (due to routing changes) and mapping updates (due, e.g., to mobility or traffic engineering) from being globally visible.

Before sending a packet, a host asks a mapping service for the corresponding locator(s) of a given identifier. The mapping service is implemented as a distributed system where mappings are stored locally, i.e., at the authorities that own the mappings. A locator encodes a loose source route[1] by specifying one possible exit point from the *CORE* area to the *INT*; one possible exit point from the *INT* to the *EDGE*; and finally the identifier of the destination host. In principle, every host can have multiple locators. According to our edge-based paradigm, the responsibility to add the locators to the packet headers is pushed to end hosts, keeping the gateway devices at the exit points simple.

[1]This does not correspond to the loose source route option of IP.

ReArch'09, December 1, 2009, Rome, Italy.

The rest of this paper is structured as follows. First we present our architecture in Section 2. Then in Section 3 we discuss its advantages. Section 4 presents a proof-of-concept implementation. Finally, we summarize related work in Section 5 and conclude in Section 6.

2. ARCHITECTURE

2.1 Overview

The motivation behind HAIR is to tackle the problem of routing table growth, to restrict the visibility of routing updates and to provide inherent support for traffic engineering and mobility. To achieve our goals, the design of HAIR follows three key ideas: (i) *Locator/identifier separation*, (ii) *hierarchical scheme* for both the routing and the mapping system that is needed to translate identifiers into locators and (iii) a hybrid *edge-based approach*. In the following, each of these principles will be explained in more detail.

2.1.1 Hierarchical scheme

Work from the past (e.g., [4, 21]) suggests that today's Internet consists of a stable "core", formed by large transit providers, and a more dynamic "edge", consisting of enterprise networks or small access providers. The typical Internet cost structure together with peering policies ensures that the number of links between an "edge" network and the "core" is limited. Most "edge" networks have a small number of upstream providers in the "core" and typically, due to costs, do not have too many upstream links to a single provider. Accordingly, we propose to group "core" networks into level 1 of a hierarchy, "edge" or *intermediate* networks into levels 2 to $n-1$, and local area networks into level n, obtaining a hierarchy consisting of n levels (see Figure 1):

***EDGE* (Level n):** This is the bottom layer of the hierarchy. Within this level routing is direct. An *EDGE* is an access network which attaches the hosts and the servers. A prototypical example is a (switched) layer-2 network such as an Ethernet LAN.

***INT* (Levels 2 to $n-1$)** An *INTERMEDIATE* (*INT*) network consists only of routers. It provides routing between attached *EDGEs* or *INTs* of the next higher level. Hence, routing tables within a *INT* need entries for all routers within the *INT*, routes to the attached *EDGEs*/*INTs*, and default routes to the *INTs* of the next higher level or the *CORE*. In the current Internet, *INTs* may correspond to access providers, enterprise networks, or content distribution networks.

CORE (Level 1): The *CORE* ensures global reachability by routing packets between the *INTERMEDIATE* networks of level 2. Routers in the *CORE* are grouped into administrative domains, each under the control of a single transit ISP. Routing tables in the *CORE* contain entries for all routers within the *CORE* and routes to the directly attached *INTs*.

According to Figure 1, the *CORE*, *INTs* or *EDGEs* hierarchy levels are connected via *attachment points*: a "level k" attachment point (L_kAP) connects a level k routing domain to a level $k+1$ one. Interconnections within the same hierarchical layer ("peerings") are also possible. We point out that the number of hierarchy levels does not have to be the same network-wide: for instance, one organization may organize its own network in two hierarchical levels, while another one may partition its network into 4 or 5 layers.

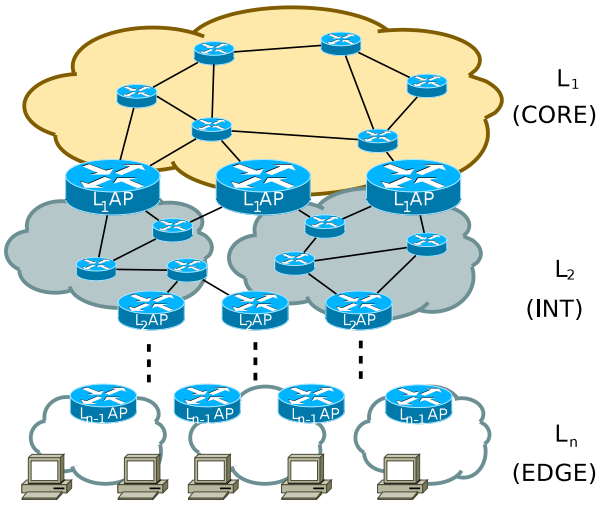

Figure 1: N-layer hierarchical network structure.

Unlike most previous proposals, HAIR's hierarchy splits routers from what is now a single AS into different levels of the hierarchy according to their function. One way to map today's Internet to a 3-layer hierarchy is to group all the routers in the backbone portion of transit ASs within the *CORE*. The policy-based BGP can be used for routing in the *CORE* area according to the policies that ISPs configure on the set of L_1APs they own. *INTs* such as enterprise networks or access provider can run their preferred routing protocol, e.g., OSPF or IS-IS. Thus HAIR preserves the autonomy of providers. Due to the limited scope of routing domains, HAIR is able to restrict the visibility of routing updates and to tackle the problem or routing table growth, see Section 3.

2.1.2 Locator/Identifier separation

Decoupling locators from identifiers provides inherent support for (end-host) mobility and avoids issues such as provider lock-in. However, a new architectural component – a *mapping service* is needed: Given a certain identifier it returns the current locator(s), see Section 2.3.

The current design of HAIR does not yet specify identifiers for end hosts. However, in the following we will tacitly assume that identifiers are organized in a global flat namespace. After all, this promises to avoid problems such as provider lock-in and renumbering.

In contrast to identifiers the namespace of locators has a local scope. They are managed by the individual *INTs*. A locator for an end host is similar to a *loose source route* from the *CORE* towards an end-host: It consists of a sequence of attachment points that need to be traversed to forward a packet from the *CORE* to the host, see Section 2.2. To support an arbitrary number of hierarchical layers, locators can have variable length. Since hosts can have multiple locators, traffic engineering, multi-homing and multi-path can easily be achieved by tweaking the mapping service.

2.1.3 Edge-based approach

Past evolution in the Internet has shown that it is easier to introduce innovation at the "edge" rather than in the "core". With HAIR, we propose an *edge-based hybrid* solution. While some lightweight functionality is added to a limited set of devices in the network (i.e., to routing domain borders), most tasks, e.g., querying the mapping system for the locator, is done by the end hosts.

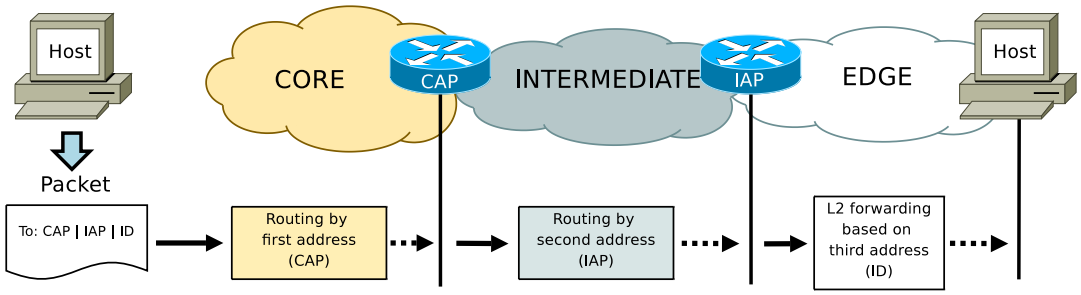

Figure 2: Packet forwarding in HAIR.

2.2 Packet delivery

Sending a packet from a source host to a destination host consists of three steps: *(i)* forwarding the packet up to the *CORE*, *(ii)* forwarding within the *CORE* and *(iii)* sending the packet from the *CORE* down to the destination host. For *(i)*, the source *INT* can either use a default route, leverage the loose source route encoded in the source locator, or combine these approaches. *(ii)* Routing within the *CORE* is based on the destination *CORE* attachment point, that is, the first address that appears in the destination locator. Finally, routing from the *CORE* towards the destination, part *(iii)*, follows the loose source route encoded in the destination locator: Whenever one of the *AP*s specified in the destination locator is traversed, forwarding is continued based on the next address in the sequence.

Figure 2 illustrates packet forwarding from source A (identifier ID_A) to destination B (identifier ID_B). We assume a 3-layer hierarchy where host A is reachable via a *CORE* AP (CAP_A) and an *INT* AP (IAP_A). A's locator then is $LOC_A = CAP_A|IAP_A$, and host B has locator $LOC_B = CAP_B|IAP_B$. First, A has to find B's identifier ID_B, e.g., via DNS. Next, it queries the mapping system to determine B's locator. The composition of $CAP_B|IAP_B|ID_B$ is the destination address of the packet. In addition, A includes its own source address, namely $CAP_A|IAP_A|ID_A$.

Using, e.g., a default route, this packet is forwarded to the *CORE* as A and B are not in the same *INT*. If A and B are in the same *INT*, i.e., if $CAP_A == CAP_B$, the next step is skipped. Within the *CORE* routing is based on the *CAP* portion of the destination address: CAP_B. Once the packet reaches CAP_B, it is handed over to the *INT*. Routing is now based on the *INT* attachment point: IAP_B. Finally, the packet reaches the *EDGE*. It is now IAP_B's responsibility to resolve the identifier, ID_B, to a layer-2 address and forward the packet to destination B.

We point out that direct "peerings" between two *INT*s, e.g. $I1$ and $I2$, can be achieved by having $I2$ export the addresses of (some of) its *CAP*s directly to $I1$, and vice versa. In this case, traffic does not need to traverse the *CORE*.

2.3 Mapping system

Our hierarchical mapping system mirrors the structure of the routing architecture. It consists of a *global directory service*, provided by the *CORE*, and a set of *Intermediate Network Mapping Service* (*IMS*s) maintained by one or a set of *INT*s. The global directory stores for all identifiers a pointer to the *IMS* which currently holds the mapping, whereas the actual mappings are kept in the *IMS*s. Since *IMS*s are administered by *INT* networks, the control over the mappings remains with the *INT*s who manage the *IMS*s.

While HAIR proposes the use of a DHT to resolve identifiers to *IMS*s, we do not put any restrictions on how to implement the *IMS* service. In principle, any distributed directory service (e.g., DNS) can be used for both the global directory at Level 1 as well as the *IMS*s. To recruit enough servers for the global directory, registries that assign resources (such as AS numbers or IP addresses) may require each AS to dedicate resources to the global DHT. Participation in the global DHT directory requires authentication and authorization by a third party, e.g., routing registries or IANA.

Our motivation for a hierarchical scheme is twofold: First, by controlling their own mappings via the *IMS* component, *INT*s are able to perform effective inbound traffic engineering. Second, the hierarchical structure of the mapping system allows us to keep some of the updates local to a single *IMS* or to a handful of cooperating *IMS*s. As an example, whenever a host moves between *EDGE*s attached to the same *INT* (e.g., a laptop moving inside an office department or between hotspots of the same provider in airports, coffee shops etc.), it is sufficient to change the mappings in the *IMS* without updating the global directory.

2.4 Dynamics

Link/Router failure: Let us presume that node v or link (v,w) inside the routing domain D, either a *INT* or *CORE*, failed. If the routing protocol inside D is able to find an alternative route between all pairs of attachment points, then no information has to be communicated beyond the routing domain D. Hence HAIR's hierarchical design ensures local visibility of routing updates. In general, even routing updates inside the *CORE* are localized in scope to the *CORE* and thus only affect the transit providers participating in the *CORE*.

Failing or unreachable attachment point: If an attachment point fails or is disconnected from the network, all locators which include it have to be updated. This implies updating portions of the mapping system. The *IMS* can monitor its attachment points and, if they change, update its bindings accordingly. Ongoing sessions can be easily handled if end hosts detect that the currently used locator is not working (see, e.g., [2]). Since *IMS*s can return multiple locators for a given identifier, after detecting that the other side's locator is not working, the end host can switch to an alternative locator.

Change of locator: Locator changes occur when a node moves to another network or when an *IMS* changes the mapping, e.g., for traffic engineering purposes. In the latter case, the updates automatically propagate to the hosts that do not have an entry in their cache or that update their cached entries. The former case differs from the attachment point failure case, as the host is usually aware of the change. Therefore a host can notify end points of active transport sessions by sending them a packet with its identifier and its new locator, as soon as it learns its new locator.

3. EVALUATION

In this section we discuss why HAIR meets the requirements for a future Internet routing system and evaluate the potential benefits of deploying it.

3.1 Requirement evaluation

Scalability of the routing system: Routing is done on a local scope inside *EDGEs*, *INTs* and *CORE*. Routing table size is mainly critical for the *CORE*. However, We expect that large ISPs in the *CORE* will have a limited number of *CAPs*, e.g., a handful for each Point of Presence (PoP). Given that the number of PoPs inside today's top tier providers is generally limited and stable over time [22], we do not expect any scalability problems with respect to routing table size. Due to the physical redundancy that is inherent to the *CORE* [14], the need for routing updates is reduced.

Scalability of the mapping system: Since each *INT* operates its own mapping service, the number of entries per *IMS* is limited. Scaling the global directory and keeping costs reasonable can be achieved by relying on a DHT. Moreover, the hierarchical design of HAIR ensures that updates to the mapping system are often kept local to one *IMS* or to a small number of cooperating *IMSs*.

Mobility: Support for mobility is inherent in HAIR due to separation of locators and identifiers. For ongoing transport sessions, a remote endpoint learns about the new locator of its counterpart as soon as the first end-to-end data packet (carrying the new locator) is received. For newly initiated sessions, the mapping system needs to be updated.

Traffic engineering (TE): By choosing which locator(s) to assign to which destination, ISPs have a new knob for traffic engineering. Each *INT* can influence where traffic enters its network by changing the locators in its *IMS*. Such updates do not need time to converge, as no routing protocol is involved. Hence, each *INT* can do inbound traffic engineering at a host granularity and prevent other hosts from interfering with its network-wide TE policies. The latter is a deployment hindrance for [18].

Easy migration: To achieve incremental deployment, we basically need to provide the *CAP* functionality for transit providers and to deploy special gateways inside each *EDGE* that translate packets, sent from legacy hosts, into HAIR packets and vice versa. For this purpose, we propose to leverage existing devices such as firewalls, gateways or proxies that already exist anyway in today's LANs. First tests with a proof-of-concept implementation of such middle-boxes are promising.

3.2 Estimation of benefits

We estimate the potential benefits to the routing system if HAIR were deployed in today's Internet. We focus on the following questions: How much can we scale the DFZ routing table? To what extent can the *INTs* isolate the *CORE* from update churn originated at the edge? For our analysis we rely on two data sources: *(i)* BGP updates and table dumps from RIPE [19], and *(ii)* classification of the ASs [4] according to their type of business into Large Transit Provider (LTP), Small Transit Provider (STP), Enterprise Customer (EC) and Content/Access/Hosting Provider (CAHP). While the former reveals information about routing table size and update churn in today's Internet, the latter allows us to distinguish "core" from

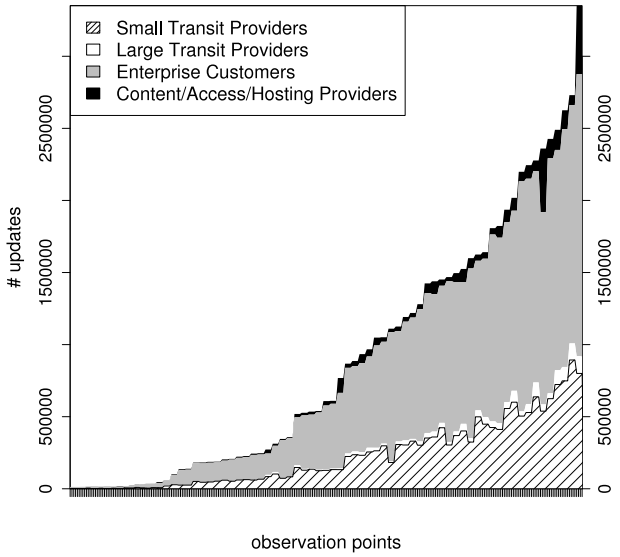

Figure 3: Number of updates per type across multiple observation points, first week of November, 2008.

"edge" ASs. Most ASs (31,704) are EC networks, followed by STP (1,663), CAHP (979) and LTP (30) domains.

To evaluate the benefits of HAIR[2] on the routing table size, we have to identify the pieces of the current Internet that would form the 3 layers. For this we assume that the *CORE* includes the backbone routers of large and small transit providers (LTPs and STPs) while CAHPs and ECs form the *INTs*. Given the 1,700 STP and LTP ASs that are then part of the *CORE* and based on the assumption that every STP/LTP operates less than 100 PoPs on average [20, 22], we estimate that the total number of locators that need to be routable inside the *CORE* is less than $1,700 \cdot 100$. As such, we now have $\approx 170,000$ locators rather than 300,000 prefixes. Overall, this suggests a considerable improvement over the current status.

Next, we study how effectively HAIR can keep updates local. For this we rely on the updates collected each year in November from 2001 to 2008. For each update trace, we check for each update if it affects a prefix originated by a LTP, STP, EC or CAHP domain. Our assumption is that an update will not be visible in the *CORE*, formed by LTP and STP domains, if it affects a prefix originated by EC or CAHP ASs.

Figure 3 shows the number of updates observed at our observation points during the first week of November 2008[3]. As some observation points only propagate updates for a small subset of prefixes, we sort the observation points along the *x*-axis according to the number of prefixes they receive. The stacked area plot partitions the number of updates, seen at each of our observation points, into the four categories LTP, STP, EC or CAHP. Hence, the *y*-value represents the total number of updates, seen at an observation point.

Figure 3 reveals that a large fraction of updates – for most observation points more than 60% – are due to prefixes originated by EC networks. This number is significantly lower for STP ($\approx$30%) and LTP networks ($\approx$5%), suggesting a considerable reduction in updates rates for *CORE* routers if HAIR were deployed in today's Internet. Since the Internet is growing much faster at the edge than in the core [4], we do not expect scalability problems.

[2]For simplicity we consider only a 3-layer deployment.

[3]The results for other years are very similar.