An Architecture for Stable, Analyzable Internet Routing*

Ramesh Govindan *Cengiz Alaettinoglu* *George Eddy* *David Kessens* *Satish Kumar*
Wee-San Lee

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292, USA

Abstract

In today's Internet, individuals, campuses and organizations obtain IP connectivity from transit providers. Internet inter-provider routing is governed by bilateral traffic exchange agreements between providers. Such independently established policies can adversely impact the stability and analyzability of Internet routing. We describe an architecture for coordinating Internet routing policies. This architecture allows providers to publish high-level specifications of their policies, and to analyze the effects of their policies on Internet routing. Several pieces of the architecture have been implemented and are in production use; we also discuss the experiences gleaned from these deployments.

Introduction

Today, spurred by the development of the World-Wide Web, many commercial entities conduct their business over the Internet. To support this, the Internet routing fabric has evolved to being commercially owned and operated. University campuses and corporate organizations purchase Internet connectivity from one or more *providers*. Each provider owns a portion of the Internet's routing fabric. Providers themselves usually interconnect at public or private *exchange points*. Collectively, providers' routers and transmission facilities enable end-to-end TCP/IP connectivity.

On its path from source to destination across the Internet, an IP packet traverses routers and links belonging to several different providers. The sequence of providers traversed by an IP packet is determined by providers' *routing policies*. Providers use routing policies to prevent some classes of traffic from traversing their facilities, or to direct their transit traffic to a preferred neighboring provider. Providers realize their routing policies by *independently* selecting, and selectively propagating, hop-by-hop routing information.

Independently established routing policy can impact the analyzability and efficiency of Internet routing. Between any two organizations, end-to-end connectivity becomes a function not just of physical connectivity, but also of intervening providers' routing policies. In a large Internet, with several thousand providers, determining the reasons for lack of connectivity can become very difficult. Moreover, un-coordinated routing policies can adversely affect application performance by resulting in the selection of longer paths than that implied by physical connectivity.

To realize ubiquitous connectivity and efficient routing, it is desirable to implement an inter-provider routing policy coordination architecture. The NSF-sponsored Routing Arbiter [9] project has been designing this architecture. This paper presents the results of the project's efforts. Our architecture consists of three elements: a

*The second author was solely responsible for the design of the Routing Policy Specification Language and the RAToolSet. This work was supported by the National Science Foundation under Cooperative Agreement NCR-9321043. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

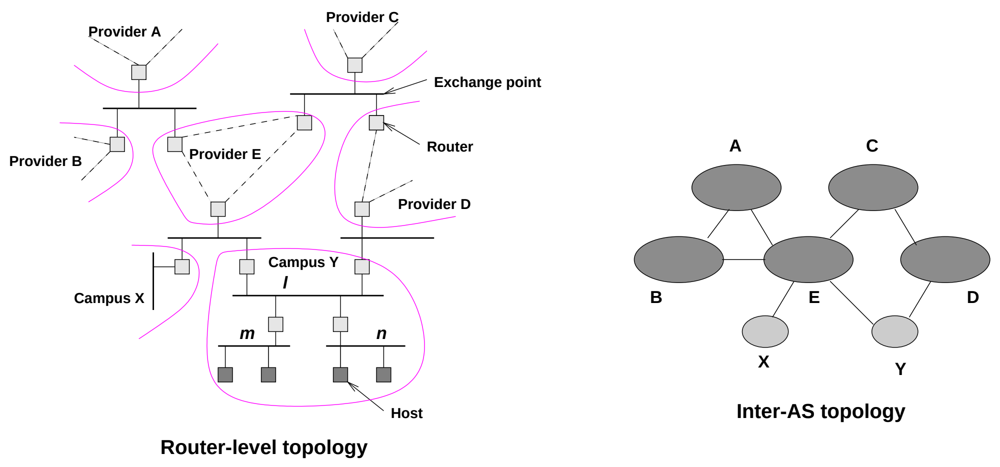

Figure 1: *The Internet's Routing Fabric:* The routing fabric physically consists of routers and links connecting hosts. It is partitioned by ownership into different Autonomous Systems (ASes). A link in the Inter-AS topology signifies traffic exchange between the corresponding ASes. Two ASs (*e.g.,* **X** and **Y**) may not be neighbors in the inter-AS topology even if they are connected in the router-level topology.

common description language for routing policy specification, a distributed registry system that allows providers to publish their policies in this language, and a suite of tools for analyzing the impact of provider routing policies on Internet traffic.

In this paper, we describe the design of this Internet routing coordination architecture, and discuss our experience with its use.

Background

The physical Internet routing fabric consists of routers and links interconnecting hosts (Figure 1). Disjoint sections of this fabric are owned and operated by different administrative entities. Each such partition of the routing fabric is called, in Internet routing parlance, an *Autonomous System* (AS). Examples of ASes include campus networks, large corporate networks and providers. An AS may cover a large geographical region; some large provider ASes even span continents. Provider ASes exchange customer traffic at Internet exchange points. An exchange point is simply a high-speed multi-access or switched communications medium connecting several ASes.

Inter-AS Topology

We can construct the *inter-AS topology* by abstracting out the router-level connectivity within ASes (Figure 1). Each link in this inter-AS topology signifies IP traffic exchange between the corresponding ASes. The structure of the inter-AS topology determines the complexity of inter-provider routing.

In the early 1990s, the North American portion of the inter-AS topology was approximately tree-structured. A single national backbone (the NSFNET) served several regional networks, from which universities and research organizations obtained IP connectivity. The NSFNET and regional networks were largely government funded entities.

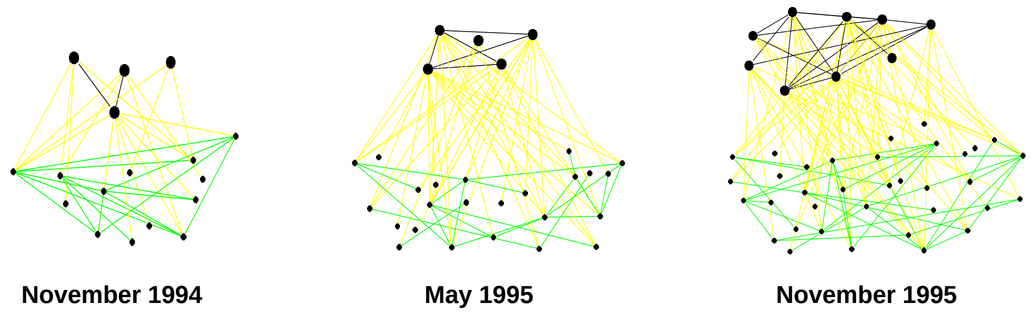

Figure 2: *Internet Inter-AS Topology:* Growth of the top two levels of the AS hierarchy and of inter-AS connectivity. For concreteness, the November 1995 picture includes eight top-level ASs, 28 second-level ASs, 16 links between top-level ASs, 41 links between second-level ASs, and 73 links between a top-level AS and a second-level AS. For various reasons, the authors have not been able to obtain more recent pictorializations of the inter-AS topology. We fully expect that the inter-AS connectivity has remained at least as complicated as that shown for November 1995.

Starting in early 1994, this three-tier hierarchical structure gradually transitioned to a more general mesh of AS interconnections. ASes were increasingly commercially owned and operated, and inter-AS connectivity was determined by bi-lateral commercial agreements on traffic exchange. Corresponding to the explosive Internet growth, the number of ASes has grown rapidly since then. Between mid 1994 and late 1995, the number of ASes grew at an astonishing rate of nearly two a day [11]. At the time of writing, there were more than 3000[1] ASes in the Internet.

Despite the reduced constraints on AS interconnections, reference [11] shows, from a study of routing table traces, that ASes can be classified by their degree in the inter-AS topology into four levels. At the top-level are the national or international backbones; examples of these include backbones operated by MCI[2], Advanced Networks and Services, and Sprint. At the next level are providers with regional or national scope. Providers with greater metropolitan area scope form the third level, and smaller metropolitan area dialup service providers typically form the fourth level. More interestingly, even though the total *number* of ASs more than doubled in the period of the trace study, the *fraction* of ASes at each level of the hierarchy appeared to have remained unchanged. That is, instead of adding more levels to the hierarchy, the AS hierarchy grew "horizontally".

This hierarchy of ASs by size, however, does not seems to have resulted in hierarchical connectivity (*i.e.,*there were many instances of top-level providers connected to level 3 or 4 providers, and so on). Figure 2 depicts the connectivity among just the top two levels of the hierarchy. Especially noticeable is the increased connectivity both within ASes of the same level, and across levels. This increase may be attributed to the emergence of several public Internet exchanges, where many tens of providers interconnect.

Inter-AS Traffic Exchange

In the inter-AS topology, each AS may have many neighbors. With each neighbor, an AS usually negotiates bilateral traffic exchange agreements. Not all ASes at an exchange point need to establish traffic exchange

[1] An AS is a routing infrastructure construct denoting an independently owned and administered portion of the Internet. Although campus networks are independently owned and administered, not all campus networks are ASes; current routing practice requires only that campuses with two or more providers be recognized as ASes. That is why this number might seem surprisingly small.

[2] Now MCI/Worldcom

agreements with each other. Thus, even though E and D in Figure 1 have routers at a common exchange point, they are not neighbors in the inter-domain topology.

What kinds of traffic exchange agreements are common in the Internet today [4]? A campus AS such as X, with a single provider E, will usually arrange to send all outbound traffic to, and receive all inbound traffic from, its provider E. Campus Y, with multiple providers E and D, may channel outbound traffic to some parts of the Internet through one provider, and to the rest of the Internet through another. In addition, Y may arrange for inbound traffic to some parts of its campus to arrive via one provider, and to the rest of its campus via the other. Neighboring providers E and C will usually arrange to exchange customer traffic; recall that a provider's customers may include other providers.

Today's Internet routing protocols can be used to realize other, more complicated, traffic arrangements. For example, E may contract with A to only carry traffic sourced by X and not by Y. Similarly, C may refuse to carry that portion of E's traffic that has transited through A. While not in widespread use today, such traffic exchange agreements could become more commonplace in the future.

Routing Policy

An AS's traffic exchange agreements with its neighbors determine how the AS routes traffic that enters or exits its infrastructure. For this reason, we say that an AS's traffic agreements determines its *routing policy*.

The Internet uses the Border Gateway Protocol (BGP [14]) to distribute the inter-AS dynamic routing information. BGP is also responsible for realizing routing policies. Conceptually, BGP works as follows: Each AS receives, from neighboring ASes, *routes* to destinations. In Figure 1, E tells C of a route to networks l, m, and n in Y through itself. From D, C might hear of a route to the same networks. C selects one of these routes, causing its traffic to destinations in Y to traverse that route. C then propagates this selection to its neighbors.

How might distribution of routes to destinations be used to realize routing policies? In BGP, this happens in one of two ways. An AS may *independently select* route information received from its neighbors. For example, C can, based on its routing policy, choose the route through D to Y, instead of the route through E[3]. In this manner, all traffic to Y from C goes through D. An AS may also *selectively propagate* route information. For example, if D does not want C's traffic to use its infrastructure to reach Y, it would not advertise to C that route to Y. It is fairly easy to visualize how BGP might be used to realize the common traffic agreements described in the previous section.

In summary, bilateral traffic agreements translate into routing policy statements that govern how routes to destinations are selected and distributed. A discussion of mechanisms for enforcing routing policies is beyond the scope of the paper.

Effects of Uncoordinated Routing Policy

In theory, an AS's routing policy is local information that is not influenced by other ASes. In practice, however, uncoordinated routing policy decisions can adversely impact Internet connectivity, routing protocol dynamics, and routing infrastructure scalability. This impact is magnified by the growth in the number of ASes and by the increasing complexity of inter-AS connectivity and traffic exchange agreements.

Since routing policies determine IP-level connectivity, two campuses may not be able to exchange IP packets even though there exists a router-level path between them [4]. In today's Internet, this lack of connectivity can reduce the potential client base of a business, or restrict academic research collaborations. In Figure 1, for

[3]Unlike traditional shortest-path routing protocols, where the path *length* determines the choice of route.
[4]To what extent this is a practical problem is not documented anywhere, to our knowledge.

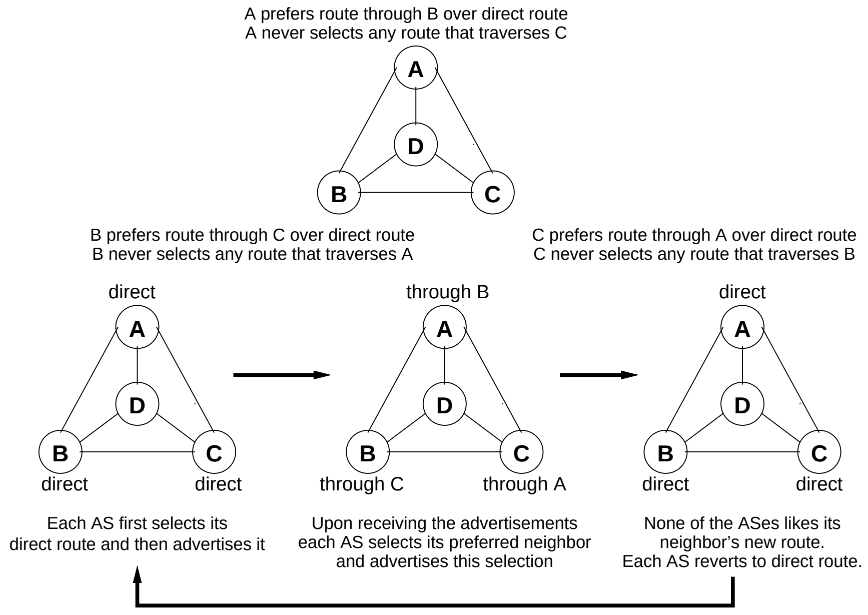
A prefers route through B over direct route
A never selects any route that traverses C
A
D
B
C
B prefers route through C over direct route
B never selects any route that traverses A
C prefers route through A over direct route
C never selects any route that traverses B
direct
through B
direct
direct
direct
through C
through A
direct
direct
Each AS first selects its direct route and then advertises it
Upon receiving the advertisements each AS selects its preferred neighbor and advertises this selection
None of the ASes likes its neighbor's new route. Each AS reverts to direct route.

Figure 3: *Non-terminating BGP route computations:* In certain topologies, ASes with mutually referential policies can result in non-terminating BGP route computation. The figure on top shows one such topology, and one choice of route selection policies. The bottom row of figures shows that, when BGP route computation is *synchronous* (that is all ASes compute and exchange routes in lock step), the route computation will not terminate.

example, even though there exists a chain of ASes connecting B and D, their customers may not be able to exchange packets. Without policy coordination among intervening ASes, it becomes difficult to determine why B and D cannot exchange packets, or how the routing policies of A, E or C may be modified to effect this packet exchange.

In the absence of policy coordination, it is possible for different ASes to establish conflicting routing policies. More specifically, three or more ASes may set up mutually dependent policies—bilateral traffic agreements that form a cycle of dependencies. These routing policies can lead to non-terminating routing information exchange in BGP. Figure 3 shows a topology, and hypothetical traffic policies, that can lead to such protocol behavior. To our knowledge, this behavior has not been observed in the Internet. Although the example in Figure 3 assumes synchronous execution, we have shown elsewhere [15] that this behavior is actually independent of the order of message exchange and route computation. Intuitively, that is because the behavior is caused by a set of policies that cannot be satisfied simultaneously.

The lack of coordination can limit the scalability of the Internet routing infrastructure as well. When Y announces to E and D a path to networks l, m, and n, that announcement contains three address entries. In IP, it is possible for Y to *aggregate* these routing entries into one, if the addresses for networks l, m and n are contiguously assigned. Other ASes can also aggregate their routing entries, enabling the routing fabric to scale better to a larger Internet. However, this aggregation cannot be performed without ensuring through coordination that the same routing policies govern these three routes [10].

Routing Policy Coordination

These examples reveal an important consequence of realizing policy through hop-by-hop routing—an AS's routing policy can affect not only its neighbors, but also other ASes in the Internet. These effects can be minimized, and Internet stability and connectivity improved, by ensuring coordinated AS policy establishment.

This paper describes an architecture for such routing policy coordination. In this architecture, each AS first accurately **specifies** its routing policy using a standard language. Then, the AS globally publishes that specification in a **routing registry**. The AS then uses this collection of routing policies through **analysis tools** to determine connectivity, to detect conflicting policies, or to discern the possibility of route aggregation. The design of this architecture is complicated by the size, geographic extent, provider diversity, and growth characteristics of the Internet.

This architecture does not require any changes to existing Internet routing and addressing. However, it relies on voluntary publication of AS routing policy. While ASes are not opposed to revealing their routing policies, they have little incentive to keep their policy specifications up-to-date. A later section describes how the design of appropriate tools can provide ASes with the right incentives.

Routing Policy Specification

The first element of our routing coordination architecture is a standard language for specifying routing policies, called *Routing Policy Specification Language* or *RPSL* [2]. Today, an AS's routing policy is implicitly specified in router configuration files of its border routers. An AS may contain several tens or hundreds of such routers; some router configuration files are known to exceed tens of megabytes. These files contain low-level directives that govern routing information exchange. The syntax and policy expressiveness of these directives differ between router vendors, or even between different releases of router software from a single vendor. Given the size of practical router configurations, and the variability in the configuration languages, publishing routing policy using router configuration languages may not achieve widespread acceptance.

Instead, we chose to design a specification language, RPSL, that enables high-level descriptions of routing policy. Such an abstract representation of routing policy is certainly feasible—AS route distribution and selection rules usually correspond to simpler, high-level, inter-AS traffic exchange predicates.

RPSL is a problem-specific object-oriented language. It defines two *classes*, the `aut-num` class and the `route` class, corresponding to the basic elements of Internet inter-provider routing. The former describes an AS's routing policy, and the latter denotes the unit of inter-AS routing information exchange. Thus, the networks l, m, and n in Y (Figure 1) would be specified as `route` *objects* (*i.e.,* instances of the route class). The neighbors of E, and its traffic exchange policies, would be specified in E's `aut-num` object.

Each RPSL object is an ordered collection of attribute-value pairs. For each object, a subset of its attributes is defined to comprise the *key*. An `aut-num` object is keyed on its name, and a `route` object is distinguished both by an IP network number (a bit-aligned initial prefix of an IP address) and by the name of the AS that originates that route. Every object is maintained by exactly one AS administrator. This maintainer is the only entity authorized to modify object attributes. An object may also refer to other objects.

Object references indicate membership in object *sets*. In RPSL, `as-sets` and `route-sets` represent unordered collections of `aut-num` objects and `route` objects respectively. Such constructs are useful for representing contiguous regions of the inter-AS topology, and the routing information entering or exiting these regions. For example, all ASes that are customers of a provider AS may belong to an `as-set` that the provider defines. That provider specifies its policies with respect to this set, rather than the individual customer ASes.

```
# Route object for l
route: l
origin: Y

# Route object for m
route: m
origin: Y

# AS object for Y
aut-num: Y
export: to D announce Y
      # announce routes originated by Y to D
export: to E announce Y
      # announce routes originated by Y to E

# AS object for E
aut-num: E
import: from X accept X
import: from Y accept Y
      # accept routes originated by X and Y
export: to C announce X
      # only announce routes originated by X to C
```

Figure 4: *Routing Policy Specification Example:* This figure shows sections of a sample RPSL specification for part of the topology described in Figure 1. In this example, routing information from Y is not propagated by E to C. Therefore, C cannot send traffic through E to hosts in Y.

In RPSL, an AS's routing policy is represented as the routing information that the AS exchanges with each neighbor. This information is specified in the `import` and `export` attributes of the AS's `aut-num` object. These attributes take as values a symbolic predicate that determines whether the AS will accept a route from, or propagate a route to, the neighbor.

Figure 4 illustrates the use of RPSL for a section of the topology described in Figure 1. In E's `aut-num` object, the simple traffic exchange policy ("accept all traffic to X from D") translates to an equivalent high-level expression ("announce to C routes to networks in$X$ "). In general, these expressions can contain unions, intersections, complements and differences of `as-sets` or `route-sets`, lending further compactness to policy representation. Though Figure 4 is sufficient to give a flavor of RPSL, a more accurate and realistic representation of policies may be found in the Appendix.

RPSL must accommodate new Internet services, such as multicast [6], and increased routing policy expressiveness. For this, RPSL defines a *dictionary* class, an instance of which specifies new attributes and their permissible operations, as well as the syntax and semantics of new objects.

The design of RPSL has been influenced by the RIPE-181 [5] routing policy specification language. The RIPE-181 language also defines classes for the basic routing constructs, ASes and routes. However, its policy expressiveness is very limited with the result that many commonly used policies cannot be specified[5]. Furthermore, it does not support sufficiently high-level abstractions and is not extensible. Until recently, routing policies were specified using extensions to the RIPE-181 language.

Now, RPSL [2] is an Internet Proposed Standard. Using tools developed by the Routing Arbiter project, several large routing registry operators have started transitioning to RPSL (May 1998). To aid this transition,

[5]For example, many providers select or discard routes based on the sequence of ASs traversed by the route (the **AS path**). RIPE-181 does not support this kind of policy specification.

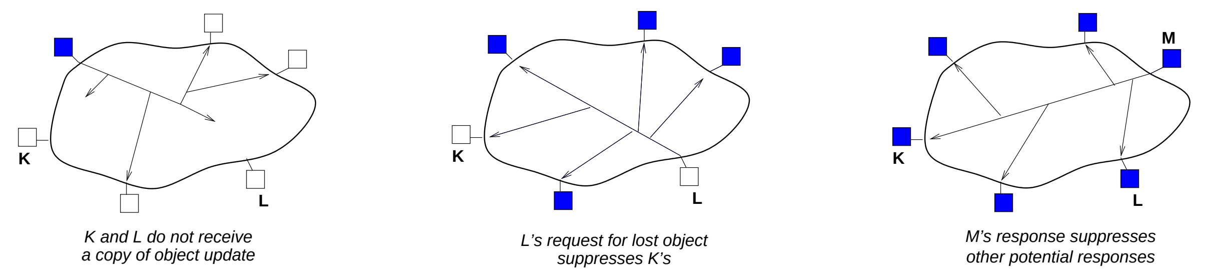

Figure 5: *RRM Principles:* RRM recovers from object losses by allowing any sub-registry to re-transmit the lost object. To reduce the possible implosion of requests and retransmissions, RRM uses timers to suppress potential requestors or respondents.

existing RIPE-181 specified policy objects were converted into RPSL using language translation software.

Routing Registry

The second component of a global policy coordination architecture is a *routing registry*, a software system that facilitates universal access to routing policies. AS administrators instantiate new RPSL objects into the registry; subsequently, analysis tools query the registry for objects.

How should a routing registry be implemented? A single, physically centralized registry does not scale well to large inter-AS topologies. It is also not desirable since access to the registry can be hindered when there are connectivity problems. A small collection of *sub-registries*—where each sub-registry presents a logically centralized view of the entire routing registry—may scale better. However, each ASs must still depend upon some other organization to manage its own data. Many providers balk at this business dependency that the solution introduces. A widely distributed registry implementation, where each AS operates a sub-registry, gives the AS more control over the availability and distribution of its policy.

In the widely distributed registry, an AS administrator registers objects in the local sub-registry. Registry users may query their local sub-registry if they wish to analyze routing policies. Some of these queries may be for *non-local* objects—objects that have been registered at some other sub-registry. Retrieving non-local objects, and authenticating their contents are two key issues in the design of the distributed registry.

Retrieving non-local objects on demand can affect the performance of analysis tools that traverse large sections of the inter-AS topology. Because we expect this to be frequent, our registry implementation fully replicates policy objects at each sub-registry. Our Reliable Registry Multicast (RRM) protocol [12] builds upon unreliable IP multicast [7] to ensure eventual replica consistency between sub-registries. Each sub-registry first multicasts object updates to all other sub-registries (Figure 5). A sub-registry that detects a loss multicasts a retransmission *request*. Before doing so, it waits for random interval drawn on the worst-case distance between any two sub-registries. This strategy probabilistically reduces the likelihood of duplicate requests; upon seeing a request for the same lost object, another requestor suppresses its own request. Any sub-registry (not necessarily the original sender) may respond; a similar suppression strategy minimizes duplicate responses. These ideas have been adapted from earlier work on Internet conferencing [8]; RRM leverages the greater delay tolerance of routing registries to implement more scalable replica consistency.

The contents of an object are digitally signed by its maintainer. The identity of the maintainer itself is authenticated by the administrator of the sub-registry where the object was registered. In turn, sub-registry administrators are (eventually) authenticated by a central certification authority (such as the Internet Assigned Numbers Authority (IANA)). By following this chain of authentication and exchanging the signatures along with the object's contents, a sub-registry can certify the contents of a non-local object [17, 16].

Each sub-registry authenticates objects, replicates them, and provides a query interface for tools. Our implementation structure reflects this division of labor. An RRM module implements the full replication of registries across the Internet. A registrar module checks local and remote objects for correct syntax, authenticates them, then dynamically builds search indices based on object key and references. Finally, a query server responds to requests from tools and humans. We use a publicly available database package (Berkeley DB [1]) for object storage management and crash recovery. This modularization has two benefits. First, the performance-critical query server is relatively well isolated from the other two functions. Second, the modules are separable so that, for example, a site may only install the registrar and the query server for testing purposes.

Our distributed registry draws significantly on experiences gained from earlier registry implementations. The first such implementation, a physically centralized registry [13], was used by the erstwhile NSFNET for routing policies of the various regional networks. Today, five large sub-registries form the Internet Routing Registry [16]; all policy objects are fully replicated at each of the sub-registries using daily inter-registry full database file transfers.

We expect to test limited deployments of our distributed registry implementation by early 1999, and have deployed a production version by the summer of that year.

Tools

The third component of the Internet routing coordination architecture is a collection of **tools** called *RAToolSet* [3]. Tools in *RAToolSet* process high-level policy descriptions. Such tools are necessary because visual inspection may not be sufficient to detect conflicting policies, for example. One reason for this is the complexity of routing policies and policy interactions. Another is that some kinds of analyses examine routing policies of a large proportion of the ASes in the rapidly growing Internet.

The tools in *RAToolSet* fall into two classes (Figure 6). **Analysis** tools examine AS policies to determine connectivity, detect policy conflicts, or discern route aggregation potential. **Management** tools simplify the maintenance of the AS's infrastructure, as we describe below. This latter class of tools induces ASes to maintain, and voluntarily publish, high-level policy descriptions.

Management tools can verify whether observed routing behavior is consistent with the intended behavior encoded in the AS's policy description. Our *route object editor (roe)* is a good example of such a tool. *Roe* takes as input the `route` objects of an AS, as well as a current routing table *dump* (*i.e.,* list of routes available to a router with that AS as the origin). It compares these routes against the published policies and points out discrepancies such as unpublished `route` objects, or incorrect route aggregation. *Roe*'s graphical user interface simplifies correcting these discrepancies. A similar tool *aoe* points out and makes it convenient to fix discrepancies in the `aut-num` objects.

A second function of management tools is the automated generation of router configurations. Today, most AS administrators manually configure their routers. Errors are often introduced during this process, for two reasons: router configuration languages contain low-level routing information exchange directives, and the configurations for different AS border routers need to be consistent but yet slightly different (for example, because different neighbors exist at different exchanges). Configuration errors have been known to have catastrophic consequences: in July 1997, a misconfigured router precipitated a prolonged Internet outage. Such errors can be reduced by automatically generating router configurations from high-level policy descriptions in RPSL; this is analogous

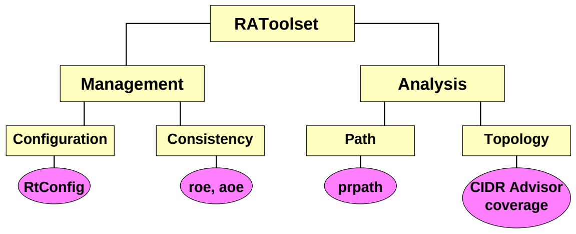

Figure 6: *A Taxonomy of Tools:* Analysis tools enable inter-provider routing coordination, but tools for managing policy descriptions are equally necessary. They provide ASes with incentive to maintain consistent and complete versions of their policy descriptions.

to programming in higher-level languages, then relying on compiler technology to generate the corresponding machine instructions. Our *router configuration generator (RtConfig)* is essentially a routing policy "compiler". It is extensively used by several Internet service providers and it can configure many popular vendor and public domain routers.

Analysis tools help ASes coordinate their routing policy to achieve Internet routing stability. *Path analysis* tools verify the existence and characteristics of connectivity between two ASes. For example, our *policy route path (prpath)* tool computes all policy-permitted paths between two ASes. *Topology analysis* tools may traverse the entire inter-AS topology. For example, our *CIDR advisor* tool lists opportunities for safe aggregation of routing information, while our *coverage* tool enables an AS to determine what sections of the Internet are not reachable from that AS. Moreover, our *Internet Routing Visualizer (IRRv)* allows users to graphically visualize the output of analysis tools. *IRRv* pictorially depicts the inter-AS topology, and supports topology navigation and abstraction.

RAToolSet has been available under freeware copyright for over three years. Its management tools, in particular, are very popular; in a recent six month period, there were nearly six hundred downloads of our tools.

Conclusions

The components of our system for Internet routing policy coordination are in various stages of development. Nevertheless, responses to our efforts have been encouraging. We expect the policy specification language to be adopted by several large providers both in US and overseas. There is widespread agreement in the Internet community on the need for a widely distributed routing registry. Many providers already use our tools for managing their production networks.

While the long-term benefits of routing policy coordination are obvious to most providers, they have been relatively slow in adopting policy specification and publication. One reason for this is the lack of an immediate *economic* value to this architecture. Our management tools, and our loosely-coupled distributed registry structure provide some of this value. Another reason is the relative infancy of the commercial Internet. Providers have not yet had compelling business reasons to specify more complicated policies—they have thus managed to operate their network without much routing coordination. As the Internet matures, we expect that the need for more complex policies will accelerate the complete adoption of this coordination architecture. Even a partial adoption will result in stable (if not completely analyzable) routing—high-level specifications and policy compilation can minimize route flux due to configuration error.

Acknowledgements

Deborah Estrin and Jon Postel significantly influenced the development of the routing coordination architecture. Our collaborators at Merit Network Inc., and several members of the U.S. and European provider community, also contributed to the design of several elements of this architecture.

References

[1] The Berkeley Database. Sleepycat Software Inc., `http://www.sleepycat.com/`.

[2] C. Alaettinoglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, and C. Villamizar. Routing Policy Specification Language (RPSL). Request for Comments 2280, Internic Directory Services, January 1998.

[3] C. Alaettinoğlu. *RAToolSet Routing Policy Analysis Tool Set*, March 1996. Available from `http://www.isi.edu/ra/RAToolSet`.

[4] C. Alaettinoğlu. Scalable Router Configuration for the Internet. *Proceedings of the 1996 International Conference on Networking Protocols*, October 1996.

[5] T. Bates, E. Gerich, L. Joncheray, J-M. Jouanigot, D. Karrenberg, M. Terpstra, and J. Yu. Representation of ip routing policies in a routing registry. Technical Report ripe-181, RIPE, RIPE NCC, Amsterdam, Netherlands, October 1994.

[6] S. Deering. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of 1988 ACM SIGCOMM Conference on Communication Architectures and Protocols*, pages 55–64, August 1988.

[7] S. Deering. Host Extensions for IP Multicasting. Request for Comments 1112, Internic Directory Services, August 1989.

[8] S. Floyd, V. Jacobson, S. McCanne, Ching-Gung Liu, and Lixia Zhang. A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing. In *Proceedings of the ACM SIGCOMM '95*, Aug 1995.

[9] National Science Foundation. Network Access Point Manager, Routing Arbiter, Regional Network Providers and Very High-Speed Services Provider for NSFNET and NREN(SM) Programs. Program Solicitation 93-52, May 1993.

[10] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy. Request for Comments 1519, Internic Directory Services, September 1993.

[11] R. Govindan and A. Reddy. An Analysis of Internet Inter-Domain Topology and Route Stability. In *Proc. IEEE INFOCOM '97*, Kobe, Japan, Apr 1997.

[12] R. Govindan, H. Yu, and D. Estrin. Large-Scale Weakly Consistent Replication using Multicast. Technical Report 98-682, Computer Science Department, University of Southern California, 1998.

[13] M. Knopper and S. J. Richardson. Aggregation Support in the NSFNET Policy-Based Database. Request for Comments 1482, Internic Directory Services, June 1993.

[14] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). Request for Comments 1771, Internic Directory Services, March 1995.

[15] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-Domain Routing. Submitted for publication.

[16] C. Villamizar, C. Alaettinoğlu, R. Govindan, and D. Meyer. Distributed Routing Policy System. Internet draft, Network Information Center, May 1998.

[17] C. Villamizar, C. Alaettinoğlu, D. Meyer, S. Murphy, and C. Orange. Routing Policy System Security. Internet draft, Network Information Center, May 1998.

A Example RPSL Specification

```
route:          128.9.0.0/16              # destination network
descr:          ISI-NET                   # name of the network
origin:         AS226                     # originating AS is 226
notify:         Prue@isi.edu              # email addr upon problems
mnt-by:         LN-MAINT-MCI              # who can modify this object
changed:        Prue@isi.edu 19950420     # who changed this object
source:         MCI                       # the repository

as-set:         AS-VERIO                  # as set name
descr:          Verio transit customers   # description of set
members:        AS2914, AS4912, AS4968,   # member ASes
                AS5738, AS-ATMNET,        # members can contain members of
                AS-ARCSTAR, ...           #  other sets
tech-c:         RB366                     # techincal contact pointer
admin-c:        RB366                     # administrative contact pointer
notify:         rw@rg.net
mnt-by:         MAINT-RGNET
changed:        heas@shrubbery.net 19981112
source:         RADB

as-set:         AS3582:AS-PEERS           # as set name
descr:          UONET's peers             # description of set
members:        AS689, AS1798, AS2914     # member ASes
tech-c:         DMM65
admin-c:        DMM65
notify:         nethelp@ns.uoregon.edu
mnt-by:         MAINT-AS3582
changed:        meyer@antc.uoregon.edu 19981112
source:         RADB

aut-num:        AS3582                    # policies of AS 3582
as-name:        UONET                     # name of AS3582
descr:          University of Oregon
import:         from AS689                # import policy for peer AS 689
                accept NOT ANY            # dont accept any routes
import:         from AS1798               # import policy for peer AS 1798
                accept AS1798             # accept routes whose objects
                                          # have AS1798 as origin
import:         from AS2914               # import policy for peer AS 2914
                accept <^AS-VERIO*$>      # accepts Verio's customers
                    AND NOT {0.0.0.0/0}   # but do not accept a default route
export:         to AS3582:AS-PEERS        # export policy for members of
                                          # as set AS3582:AS-PEERS
                announce AS3582           # announce our routes
```

```
admin-c:        DMM65
tech-c:         DMM65
notify:         nethelp@ns.uoregon.edu
mnt-by:         MAINT-AS3582
changed:        meyer@antc.uoregon.edu 19980128
source:         RADB
```

BIOGRAPHIES

Ramesh Govindan Ramesh Govindan is Project Leader for the Routing Arbiter Project at the Information Sciences Institute and a Research Assistant Professor in the Computer Science Department at the University of Southern California. He received his B. Tech. degree at the Indian Institute of Technology, Madras, in 1987 and his M.S. and Ph.D. degrees from the University of California, Berkeley, in 1989 and 1992 respectively. His research interests are in computer networking and operating systems.

Cengiz Alaettinoglu Cengiz Alaettinoglu received the B.S. degree in Computer Engineering in 1988 from the Middle East Technical University, Ankara, and the M.S. and the Ph.D. degrees in Computer Science in 1991 and 1994 from the University of Maryland, College Park. He is a Computer Scientist at the Information Sciences Institute and a Research Assistant Professor in the Computer Science Department, University of Southern California. His research interests are in computer networking, distributed systems, and operating systems. His current work is on inter-domain routing protocols with type-of-service and policy constraints for large and heterogeneous internetworks.

George Eddy George Eddy has worked in the area of computer systems and networks for the last ten years. His work has involved distributed application development, real-time embedded systems, network protocol development and testing, and network management. George received a BSCS from the California State University, Long Beach and a MSCS from the University of Southern California, where his focus was computer networks and multicast routing.

Satish Kumar Satish Kumar is currently undertaking a Ph.D. in Computer Science at the University of Southern California. He received a B.Tech in Electronics and Communication Engineering from the Indian Institute of Technology in Madras, India in 1994 and a M.S. in Computer Engineering from the University of Southern California in 1996. His research interests include unicast and multicast routing, network simulation tools and networks of embedded devices.

WeeSan Lee WeeSan Lee has been working on Routing Arbiter Project in USC/ISI for more than two years. His work has involved tools development, maintenance and software release. He has worked on database, security, web cache and multicast protocol before. Lee received his BSCS from the Fu-Jen University in Taiwan and his MSCS from the University of Southern California focusing on Software Engineering and Computer Network.

Contact Information

Corresponding author:

Ramesh Govindan
USC/Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina Del Rey, CA 90292
Phone: +1-310-822 1511 (x103)
Fax: +1-310-823-6714
Email: govindan@isi.edu

Other Authors:

Cengiz Alaettinoglu
Email: cengiz@isi.edu
George Eddy
Email: eddy@isi.edu
David Kessens
Email: davidk@isi.edu
Satish Kumar
Email: kkumar@isi.edu
WeeSan Lee
Email: wlee@isi.edu