Scalable Router Configuration for the Internet*

Cengiz Alaettinoglu†
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
cengiz@isi.edu

Abstract
Configuring routers to realize various policies is a difficult task. We classify the typical Internet routing policies and present a specification language where these policies can be specified at high level. We also present a tool that analyses the policies specified in this language and generates the low level router configurations.

1 Introduction
Internet is an interconnection of networks. These networks are administered by different organizations. The set of networks administered by the same organization is called an autonomous system (AS). Reaching a destination in another AS requires the use of resources of other ASes along the route. This imposes policy constraints on the inter-AS traffic. For example, an AS may only allow traffic from certain ASes to use its resources, or may prefer to use the resources of a particular AS over other ASes.

These policies are realized by the inter-AS routing protocol, BGP[5], and need to be configured in the routers. This task is both tedious and not straightforward. First, each AS has a large number of routers that need to be configured. Second, these configurations can be very large (of the order of hundreds of thousands of lines). Third, different routers need to be configured consistently. Fourth, an AS may have routers from different vendors, and each vendor provides different mechanisms and syntax for configuration. Fifth, misconfiguration can have severe impact; it can lead to non-optimal routing, it can lead to certain destinations becoming unreachable, or can even partition the Internet.

In this paper, we address the problem of scalable router configuration. We present a language, Routing Policy Specification Language (RPSL)[2], to represent policies at a high level, and a tool, RtConfig, to generate low level router configurations from these specifications.

2 Typical Policies in the Internet
An organization connects to the Internet by buying services from a Network Service Provider (NSP). NSP networks vary in size; most of these first level NSPs that the organizations connect operate in a single metropolitan area.

To get connected to the global Internet, the NSP in turn buys services from another NSP. This second level NSP is usually geographically more spread out and most of its customers are other NSPs. The second level NSPs are often called regional NSPs.

The regional NSPs in turn buy services from other NSPs, referred to as backbone NSPs. Backbone NSPs cover a very large geographical area, often a continent, and often have many intercontinental connections. Backbone NSPs are fully connected to each other to provide global connectivity.

The policies in the Internet reflect these provider-customer relationships. It is almost always the case that an NSP only allows its resources to be used only by its customers (or to reach its customers). In the rest of this section we classify typical Internet policies.

Provider-Customer Policies In this classification, we examine the policies of a customer A which is connected to one provider B. The customer is often interested in connecting to the entire Internet through the provider. The NSP’s policies are symmetrical; it allows the customer to use its resources to reach any site on the Internet, and allows all sites on the Internet to use its resources to reach the customer. If the customer is itself an NSP (i.e. A has its own customers) then the A’s customers are also treated the same way as A; that is NSP’s resources can be used by them, as well as by others to reach them.

Provider-Provider Policies In this classification, we examine the policies of providers A and B that are connected to each other without buying services from each other. A and B often connect in this way so that their customers can reach each other efficiently without going through a third provider. However, A and B almost always do not allow the customers of the other provider to use its resources to reach destinations other than its customers.

*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(s).
†Also with Computer Science Department.
Multi-Homing Policies

In this classification, we examine the policies of a customer AS A which is connected to provider ASes B and C. Often, the customer uses one of the providers as the primary connection to the Internet, and if that connection fails, it uses the other provider. Or it may use both providers simultaneously for load balancing. Note that the customer does not allow B and C to reach each other over its network.

3 Routing Policy Specification Language (RPSL)

RPSL is based on an earlier policy language known as RIPE-181 [3]. Many of the policies exercised today cannot be expressed in RIPE-181. For example, it is not possible to specify policies of Figures 7 and 9.

In RPSL (and in RIPE-181), the policies are specified in policy objects. For example, each route is specified in a route object and each AS is specified in an aut-num object. route-set and as-set objects are used to make sets of route and aut-num objects. The policies are expressed as relationships between these objects. These policies are registered in the Internet Routing Registry (IRR).

We next present these objects. We focus on the scalable router configurations aspect, and skip many important but not relevant aspects of these objects.

3.1 Route Objects

A route object is registered for each address prefix routed in the Internet. The route object documents the address prefix, and the AS which originates it. Figure 1 shows some example route objects.

The collection of route objects registered in IRR with a particular AS number in the origin line defines the set of routes originated by that AS. Assuming the two route objects shown in Figure 1 were the only route objects registered in IRR, they define the set of AS 226’s routes namely {128.9.0.0/16, 128.99.0.0/16}.

3.2 Route Set Objects

The route set objects define arbitrary route sets and gives each set a name. Figure 2 shows an example which defines the route set RS-RESERVED ROUTES. RS-RESERVED ROUTES contains the routes 0.0.0.0/0, 205.197.37.0/24, and 205.197.38.0/24.

3.3 Autonomous System Objects

The autonomous system objects describe the policies of ASes. Each organization registers an autonomous system object in the IRR for its AS. Figure 3 shows the autonomous system object for AS 226 (Los Nettos). In the example, the policies toward neighbor AS 2150 are specified. The as-in line specifies the policies for importing routes and the as-out line specifies the policies for exporting routes. In this example, AS 226 imports only one route from AS 2150, namely 204.70.1.0/24, whereas it exports all of its routes to AS 2150 ("announce AS 226" means to export the set of AS 226’s routes).

Note that if AS 226 grows and originates another route, say 128.98.0.0/16, it registers a route object for 128.98.0.0/16 with origin AS 226. AS 226’s aut-num object does not change, since it refers to the set name AS 226, instead of each route explicitly.

On the other hand if AS 226 decides to import one more route from AS 2150, it needs to update its aut-num object, since in this case it chose to specify the set of routes explicitly.

3.4 Autonomous System Set Objects

Autonomous system set objects are used to group autonomous system objects. Figure 4 shows an example which defines a set named AS-226-CUSTOMERS which contains AS 226, AS 2150, and AS 31.

3.5 Specifying Policies using RPSL

We next illustrate how these objects are used to specify typical Internet policies.

Provider-Customer Policies

Customer imports all the routes that the NSP has and exports its routes to the NSP. The NSP’s policies are symmetrical: it exports all routes that it has to the customer, and it imports from the customer only the customer’s routes. Figure 5 illustrates one way of expressing these policies using RPSL where AS1 is the provider and AS 2 is the customer.

Figure 1: Route Objects

Figure 2: Route Set Object

Figure 3: Autonomous System Object

Figure 4: Autonomous System Set Object
aut-num: AS1
as-out: to AS2 announce ANY
as-in: from AS2 accept AS2

aut-num: AS2
as-out: to AS1 announce AS2
as-in: from AS1 accept ANY

Figure 5: Provider-Customer Policies in RPSL

as-set: AS-2-CUSTOMERS
members: AS2 AS20 AS200, ...

aut-num: AS1
as-out: to AS2 announce ANY
as-in: from AS2 accept AS-2-CUSTOMERS

aut-num: AS2
as-out: to AS1 announce AS-2-CUSTOMERS
as-in: from AS1 accept ANY

Figure 6: Provider-Customer Policies in RPSL

In the example, "announce ANY" means export any route that AS 1 has, "accept AS2" means accept only AS 2’s routes. Policies of AS 2 are symmetrical. Note that if AS 2 adds or deletes route objects, there is no need to update the aut-num objects. The addition and deletion of route objects will implicitly update AS 1’s and AS 2’s policies, and affect the router configuration files.

If the customer is itself an NSP, i.e., it has its own customers, the set of routes passed to the provider includes its customers’ routes as illustrated in Figure 6. In this example, "accept AS-2-CUSTOMERS" means that for each AS X in AS-2-CUSTOMERS accept AS X’s routes.

If AS 2 gets a new customer, say AS 2000, AS 2 updates the definition of the AS-2-CUSTOMERS. The policies specified in the aut-num objects do not change. Similarly, if one of the customers of AS 2 had more routes, it would register new route objects, which indirectly update AS 1’s and AS 2’s policies without changing the definition of aut-num or as-set objects.

Provider-Provider Policies In this case, the policies of both providers are to export only their customer routes to the other provider, and to import only the customer routes of the other provider. Figure 7 illustrates how this is expressed using RPSL where both AS 1 and AS 2 are providers. In this example, we chose to specify filters based on the AS_PATH attribute: the filter "<AS-2-CUSTOMERS>$" matches routes whose AS_PATH attribute ends with an AS in AS-1-CUSTOMERS.

as-set: AS-1-CUSTOMERS
members: AS1 AS10 AS100, ...

aut-num: AS1
as-out: to AS2 announce <AS-1-CUSTOMERS>
as-in: from AS2 accept <AS-2-CUSTOMERS>

aut-num: AS2
as-out: to AS1 announce <AS-2-CUSTOMERS>
as-in: from AS1 accept <AS-1-CUSTOMERS>

Figure 7: Provider-Provider Policies in RPSL

Multi-Homing Policies In this case, both providers export all the routes that they have to the customer. The customer imports these routes and assigns a preference to each route imported. The customer exports only its own routes to both providers. It does not export the routes that it imports from the providers. If it did, the customer could become transit between these two providers. Providers import from the customer only the customer’s routes. Figure 8 illustrates how this is expressed using RPSL where AS 1 is the customer and buys services from providers AS 2 and AS 3. In this example AS 1 prefers to use AS 2 over AS 3.

as-set: AS-1-CUSTOMERS
members: AS1 AS10 AS100, ...

aut-num: AS1
as-out: to AS2 announce AS1
as-out: to AS3 announce AS1
as-in: from AS2 action pref = 1 accept ANY
as-in: from AS3 action pref = 2 accept ANY

aut-num: AS2
as-out: to AS1 announce ANY
as-in: from AS1 accept AS1

aut-num: AS3
as-out: to AS1 announce ANY
as-in: from AS1 accept AS1

Figure 8: Multi-Homing Policies in RPSL

Policies Towards a Set of Autonomous Systems Often an NSP has tens to hundreds of customers. The policy toward each of these customers are similar; that is for customer X, export all routes to X, and import X’s routes from X. Similarly, an NSP may buy services from multiple higher level NSPs and a similar generalization of policies may apply. Figure 9 illustrates how these generalized policies can be specified using RPSL. We only show the policies of AS 1. In
as-set: AS-1-CUSTOMERS
members: AS1 AS10 ...

as-set: AS-1-PROVIDERS
members: AS2 AS20 ...

as-set: AS-1-OTHERS
members: AS3 AS30 ...

aut-num: AS1
as-out to AS-1-CUSTOMERS announce ANY
as-in from AS-1-CUSTOMERS accept PeerAS
as-out to AS-1-PROVIDERS announce AS-1-CUSTOMERS
as-in from AS-1-PROVIDERS accept ANY
as-out to AS-1-OTHERS announce AS-1-CUSTOMERS
as-in from AS-1-OTHERS accept AS-PeerAS-Customers

Figure 9: Policies Towards an AS Set

this example,

\[
\text{as-in: from AS-1-CUSTOMERS accept PeerAS}
\]

is equivalent to specifying

\[
\text{as-in: from AS1 accept AS1}
\]

\[
\text{as-in: from AS10 accept AS10}
\]

The keyword “PeerAS” stands for the AS number from/to which this policy applies.

Again when AS 1 connects to another AS, all it has to do is to include that AS in the correct sets.

4 Configuring Routers using RtConfig

To generate low level router configurations from the high level policies specified in RPSL, we developed a tool called RtConfig. RtConfig supports many public domain and vendor router configuration formats. RtConfig is written in C++. It has been publicly available on the Internet since late 1994. It is currently being used to configure routers by many NSP’s throughout the world. We next illustrate how to use RtConfig to generate low level router configurations.

For each router, an AS prepares a template file for RtConfig. The template file is just like a router configuration file, except the policy configuration lines are removed and replaced with RtConfig commands. RtConfig reads this file line by line, and prints the lines it reads, except for the lines that start with orRtConfig. These lines instruct RtConfig to perform special operations. An example template file is shown in Figure 10. In this example, the command “orRtConfig import AS1 198.32.4.1/32 AS2 198.32.4.25/32” instructs RtConfig to generate low level import policies where the router 198.32.4.1 in AS 1 is importing routes from router 198.32.4.25 in AS 2. The other orRtConfig commands instruct the RtConfig to use certain names and numbers in the output that it generates.

5 Conclusion

In our approach, instead of specifying policies toward each route or each AS, we formed sets of routes and ASes and specified policies as relationships between these sets. Hence, as Internet grows, that is as more routes and ASes are introduced, we reduced the task of router configuration to the task of updating set memberships.

These objects are registered by different organizations in IRR (i.e. AS 1 registers its own routes, AS 2 registers its own routes, etc.). The objects registered in IRR by an organization are used to generate router configurations of any AS, not just the AS which registered these objects. For example, whenever a customer AS registers new route objects, the router configurations of the customer AS as well as its provider ASes (and perhaps other ASes) are updated automatically. In a way, ASes share the router configuration tasks of each other to achieve scalable router configurations.

RtConfig [1] is the first tool which can automatically generate complete router policy configurations from an higher level policy specification. rlc [4] (route list compiler) is another effort at configuring routers. It is cisco specific and can only generate access lists, a small piece of the router configuration task.

References