Improvement of Consistency among AS Policies on IRR Database

Masasi Eto*, Youki Kadobayashi*, Suguru Yamaguchi*

* Nara Institute of Science and Technology, Ikoma-shi, 630-0101 Japan.
{masash-e, youki-k, suguru}@is.aist-nara.ac.jp

ABSTRACT

This paper presents an architecture which investigates the consistency of AS policies in the whole Internet Routing Registry (IRR) databases in the world. We also propose a system to prevent the increase of inconsistency. Since inconsistency hampers connectivity between the ASes, the consistency of IRR databases are crucial for stable operation of the Internet. Through investigations, we have observed that significant portion of AS policies are either incorrectly specified or outdated. Based on this observation, we propose and implement a system that detects these inconsistencies and notifies the operators to correct them. Finally, we evaluate this system.

1 Introduction

Border Gateway Protocol (BGP) is crucial to the overall reliability of the Internet [1], [2], but faults in BGP have been known to disrupt large regions of the Internet.

For example, in April 1997, AS7007 accidentally announced routes to most of the Internet and disrupted the connectivity for over two hours [3]. In April 2001, AS3561 propagated more than 5000 improper route announcements from one of its downstream customers, again led the global connectivity problems [4]–[6].

To alleviate this problem, Internet Routing Registry (IRR) is expected as a tool which increases efficiency of operation on BGP network. IRR holds policies written in Routing Policy Specification Language (RPSL) [7] that are registered by operators of AS. These policies consist of information such as the AS numbers, the maintainer of the AS, and the routing policies.

However, operators generally view the IRR as an obscure and difficult system rather than an useful tool for network operations. This understanding prevents the widespread use of the IRR.

One of the reasons of this skepticism ascribes to inconsistency of IRR database. Router configurations generated from IRR database cannot be trusted because they may contain inconsistencies which make the communication between the ASes impossible.

In this research, we have investigated the consistency of AS policies in the whole IRR databases: 55 IRR servers in the world such as RIPE, RADB and IRRs mirrored by RADB. As a result of investigation, we have obtained a key finding that the significant portion of the AS policies are either incorrectly specified or outdated. Based on this result, we propose a change to the network operations that would eliminate most of the inconsistency we observed. Our proposal aims at stable and less labor-intensive Inter-domain routing with the IRR.

2 IRR

IRR is the global Internet resource database that stores routing policy such as the AS number and the prefix information. IRR consists of several objects (Route Object, Aut-num Object, Maintainer Object, etc.). Policy registered in IRR is written in Routing Policy Specification Language (RPSL). RPSL is designed to describe the specific routing information by import and export sentences in Aut-num object. Operators can generate the vendor specific router configuration from the policy data [8].

However, in current IRR operation, IRR contains the inconsistency in its database. IRR database is built up by each maintainer of ASes registering policy about their own AS. If we accumulate policies to IRR database in this way, it will contain many inconsistencies. As a result, when we generate the router configurations from this database, the connectivity between peering ASes will be lost. Otherwise, the link selection which operator doesn't intend will occur.

3 Classification of the inconsistency

In this section, we discuss about the classification of inconsistencies that would prevent the connectivity between peering ASes. Inconsistencies are roughly classified into the following two types :

- inconsistency in routing information import
- inconsistency in routing information export

In the following subsections, we explain each type of inconsistencies. Then we elaborate classified inconsistencies.

3.1 Inconsistency in routing information import

If an AS expects to establish the connectivity with some ASes and configures to import their routing information, and if the peering AS doesn't export their routing information to

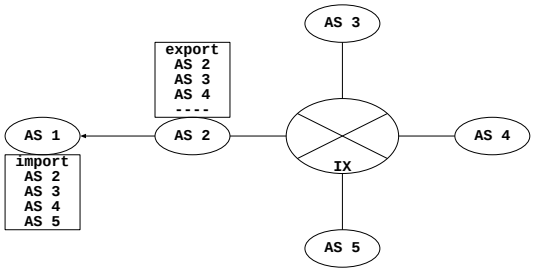

Figure 1: Inconsistency in routing information import

```
aut-num: AS 1
as-name: EtoNet
...

import: from AS 2
        accept AS 2, AS 3, AS 4, AS 5
...
```

Figure 2: Policy registered by AS 1

the AS, the AS can't get the connectivity to those ASes. We explain this case of problem as the *Inconsistency in routing information import* in the following example.

As shown in Figure 1, AS 1 and 2 are under contract that AS 2 provides the transit to the traffic from AS 1 to 5, 4 and 3. According to this contract, the operator of AS 1 registered the policy shown in Figure 2 which is configured to import the route of AS 2, 3, 4 and 5 from AS 2. Otherwise the policy registered by the operator of AS 2 is shown in Figure 3 and in the policy, route of AS 5 is missing by accident.

Router configurations generated from these policies by RtConfig are shown in Figure 4 and 5. If the operators commit these configurations to their routers as they are, router of AS 1 couldn't receive the route of AS 5 so that AS 1 couldn't establish connectivity with AS 5.

3.2 Inconsistency in routing information export

If the peering AS configures to export the expected routing information, and if the AS doesn't import their routing

```
aut-num: AS 2
as-name: SaiNet
...

export:  to AS 1
         announce AS 2, AS 3, AS 4
...
```

Figure 3: Policy registered by AS 2

```
import proto bgp as AS 2 {
  192.168.2.0 masklen 24 exact;
       //route information of AS 2/
  192.168.3.0 masklen 24 exact;
       //route information of AS 3/
  192.168.4.0 masklen 24 exact;
       //route information of AS 4/
  192.168.5.0 masklen 24 exact;
       //route information of AS 5/
  all restrict;
}
```

Figure 4: Configuration on a router in AS 1

```
proto bgp aspath .* origin any {
  192.168.2.0 masklen 24 exact;
       //route information of AS 2/
  192.168.3.0 masklen 24 exact;
       //route information of AS 3/
  192.168.4.0 masklen 24 exact;
       //route information of AS 4/
  all restrict;
}
```

Figure 5: Configuration on a router in AS 2

information from the peering AS, the AS can't establish the connectivity with those ASes. In the following example, we describe this case of problem as the *Inconsistency in routing information export*.

AS 1 and 2 registered the policies shown in Figure 7 and 8. In this case, AS 2, the transit provider registered the proper policy according to the contract. However, in the policy of AS 1, sentence to import the route of AS 1 is missing by accident. The router configurations generated from these policies are shown in Figure 9 and 10. As a result, AS 1 couldn't establish the connectivity with AS 5.

Based on these premises, we classified these inconsistencies in more detail Table 1.

4 Methodology

In this research, we aim to establish a mechanism to conduct the exact inspection on the global scale with appropriate

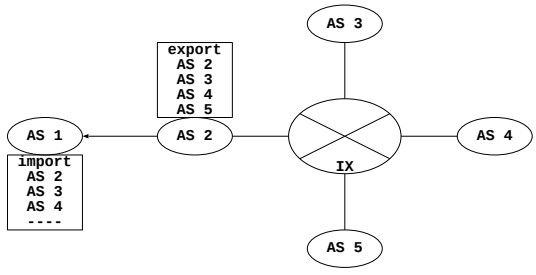

Figure 6: Inconsistency in routing information export

Table 1: Classification of inconsistencies

Inconsistencies in routing information import	peering AS-SET doesn't exist on IRR database
	peering AS doesn't exist on IRR database
	peering AS doesn't export any routes to the AS
	peering AS doesn't export route which the AS imports
Inconsistencies in routing information export	peering AS-SET doesn't exist on IRR database
	peering AS doesn't exist on IRR database
	peering AS doesn't import any routes from the AS
	peering AS doesn't import route which the AS exports

```
aut-num: AS 1
as-name: EtoNet
…
import:  from AS 2
         accept AS 2, AS 3, AS 4
…
```

Figure 7: Policy registered by AS 1

```
aut-num: AS 2
as-name: SaiNet
…
export:  to AS 1
         announce AS 2, AS 3, AS 4, AS 5
…
```

Figure 8: Policy registered by AS 2

efficiency. We defined the following methodology.

At first, we propose and implement a system to investigate the consistency of AS policies in the whole IRR databases in the world. Second, we propose and implement a system to prevent increasing such inconsistency. Third, we analyze the results of the investigation and perform qualitative evaluation of the implementation.

4.1 Aggregation of IRR databases

We decided to collect and aggregate all policies of IRR database in the world because of the following reason.

When an organization gained AS number from the Regional Internet Registry (RIR), each organization needs to register their policy to the IRR database. However, the IRR server is managed by the arbitrary organizations in the world. Therefore, the policies of each ASes are distributed to each of the IRRs. To inspect the consistency of AS policies, we have to collect and store them in one place. The database is opened to the public by FTP, so that we decided to collect the 52 IRR databases including RIPE, RADB and APNIC. In this paper, we call the collected databases as *Unified IRR Database*.

```
import proto bgp as AS2 {
  192.168.2.0 masklen 24 exact;
       //route information of AS 2/
  192.168.3.0 masklen 24 exact;
       //route information of AS 3/
  192.168.4.0 masklen 24 exact;
       //route information of AS 4/
  all restrict;
}
```

Figure 9: Configuration on a router in AS 1

```
proto bgp aspath .* origin any {
  192.168.2.0 masklen 24 exact;
       //route information of AS 2/
  192.168.3.0 masklen 24 exact;
       //route information of AS 3/
  192.168.4.0 masklen 24 exact;
       //route information of AS 4/
  192.168.5.0 masklen 24 exact;
       //route information of AS 5/
  all restrict;
}
```

Figure 10: Configuration on a router in AS 2

4.2 Database Synchronization

We defined the cycle of updating the Unified IRR Database as once a day.

The Unified IRR database is required to be up-to-date. Meanwhile, IRR Databases keep their consistency by mirroring each other once a day in current style of operation. Namely, it takes a day at the maximum for the latest entries of the one IRR database to be mirrored by the other IRR databases.

Based on this standpoint, we assume that the daily update of the Unified Database is appropriate.

5 Proposed system and implementation

To investigate and prevent these inconsistencies of the IRR databases, we propose *Policy Check Server* which consists of three main components as follows.

- To inspect the consistency between ASes, we need all

policies of whole IRR databases in the world. Therefore, we construct a common database called *Unified IRR Database* which includes all policies by *DBGenerator*.

- *Database Checker* inspects how many inconsistencies exist on Unified IRR database.

- *Policy Checker* inspects whether the policy which the operator of AS is about to register is consistent with the policies of peering ASes. Then we make clear the necessity of inspecting the consistency of the policies.

5.1 DBGenerator

DBGenerator extracts the import sentences and the export sentences from the collected policies and creates AS objects which hold each value in AS's policy. Then DBGenerator injects the AS objects into the database which is constructed by PostgreSQL database.

5.2 Database Checker

Database Checker inspects how many inconsistencies exist on IRR database. Therefore, we make clear the necessity of Policy Check Server which assures the policy is consistent with other policies. Database Checker inspects the whole policies in IRR database according to the algorithm shown in Figure 11.

The algorithm consists of three phases.

1. Database Checker specifies the peering AS by the import or export sentences and hold the peer AS as an AS object. If information about the peering AS doesn't exist on IRR database, Database Checker flags the information as an inconsistency.

2. Database Checker compares the import sentences of input AS and the export sentences of the peering AS. If the peering AS doesn't have export sentence which specifies input AS as a peer like this :

   - export: to *input AS* accept AS 3 ...

   Database Checker collects the information as an inconsistency. Otherwise, Database Checker checks whether the peering AS exports the route prefix which the AS intends to import from. If it doesn't, Database Checker collects the information as an inconsistency. In the next phase, Database Checker compares the export sentences of input AS and the export sentences of the peering AS.

3. Database Checker outputs the collected inconsistencies to the log file.

```
[ specify peering AS ]
    extract import, export sentences from input AS object ;
    if (the peering AS (or AS-SET) exists on database) {
        create "Autnum" object as a peering AS ;
    } else {
        warn as an inconsistency ;
    }
[ inspection of import sentence ]
    for (number of import sentence, repeat following processes) {
        for (number of export sentence, repeate following processes) {
            if (the export sentence specify input AS as a peer) {
                if (the sentence doesn't export required routes) {
                    warn as an inconsistency ;
                }
            } else {
                warn as an inconsistency ;
            }
        }
    }
[ inspection of export sentence ]
    for (number of export sentence, repeat following processes) {
        for (number of import sentence, repeate following processes) {
            if (the import sentence specify input AS as a peer) {
                if (the sentence doesn't import required routes) {
                    warn as an inconsistency ;
            } else {
                warn as an inconsistency ;
            }
        }
    }
```

Figure 11: Algorithm of inspection

5.3 Policy Checker

Policy Checker gives the operator an opportunity to inspect the policy which he intends to register to IRR database. Policy Checker keeps all of the latest entries of IRR database. So it is suitable for Policy Checker to be managed inside IRR server.

The process is as follows.

1. The policy input by the operator is transmitted to Policy Checker which is deployed as a Java Servlet on Apache (WWW server) and Tomcat (Web application server).

2. Policy Checker creates an AS object from input policy and transmit it to Database Checker.

3. Then Database Checker inspects the consistency between input policy and peer AS's policy as described in section 5.2.

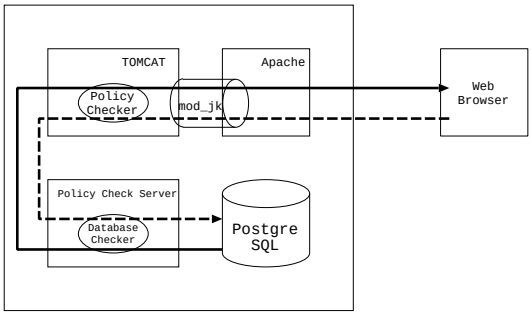

Figure 12: Basic components of policy checker

4. Database Checker returns the collected information of the inconsistency to Policy Checker.

5. Policy Checker generates HTML document from the result of the inspection and displays it to the operator's web browser.

The process of inspection is given with the web-based interface. If any inconsistencies are detected, Policy Checker notifies the warnings on the operator's web browser. Then the operator of the AS may correct the corresponding entries and register the consistent policy. The basic composition is shown in Figure 12.

5.4 Consistency Chain

By correcting the inconsistencies between peering ASes, it is possible to exchange the route information between ASes that are not directly peering. Eventually it is also possible to improve the consistency of the whole IRR databases. We describe this mechanism as follows.

For example, we use the situation shown in Figure 13. In this situation, the requirement is to give AS 1 a connectivity to AS 2 and AS 3. To complete this requirement, each AS has to declare to import or export expected routes.

1. At initial state (Figure 13(a)), AS 2 doesn't export the route of AS 3 to AS 1. Furthermore, AS 3 doesn't export the route of AS 3 itself to AS 2. At this state, route of AS 3 is never transmitted to AS 1 so that AS 1 can't establish the connectivity to AS 3.

2. At this state, if the operators use Policy Checker, it tells them that AS 2 doesn't export the expected route to AS 3. So that the operator of AS 2 would be able to correct the corresponding entry (Figure 13(b)).

3. However at next state (Figure 13(c)), Policy Checker tells the operator of AS 3 that AS 3 doesn't export any routes to AS 2. Based on this warning, the operator of AS 3 would be able to add entry properly.

4. As a result, the policies of each AS are corrected and AS 1 would be able to get the route of AS 3 (Figure 13(d)).

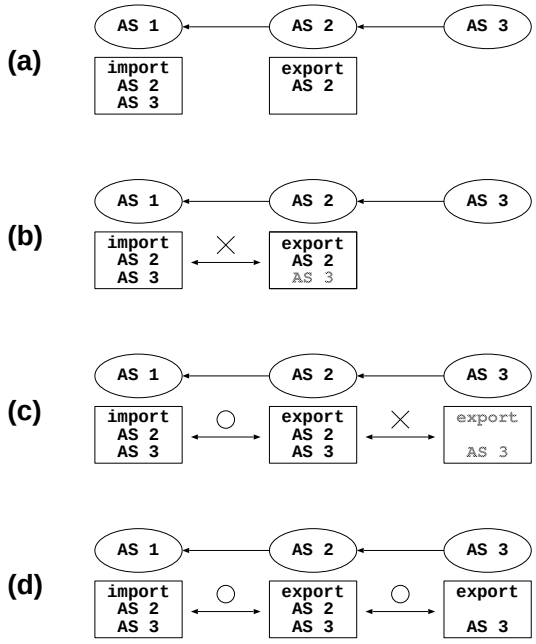

Figure 13: Consistency chain

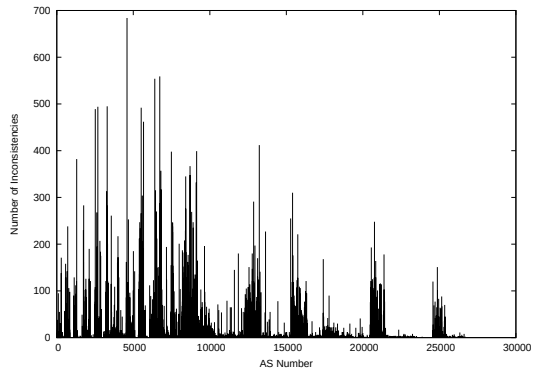

Figure 14: Number of inconsistencies of each AS

Finally, connectivity between AS 1 and AS 3 is established. Thus, by using Policy Checker, it is possible to check consistency between ASes that are not directly peering.

6 Analysis of inspection result

As a result of the investigation, we have found that 55.8% of AS has at least one inconsistency categorized in Table 1 on IRR databases. More detailed result is shown in Figure 6. Figure 6 tells us that there is the deviation in the number of inconsistencies by the AS number. In other words, inconsistency increases as the number of AS becomes larger. We assume that the AS with a smaller AS number tends to have many peers so that the AS has many import or export sentences and many inconsistencies.

We divided this result into two groups which have large AS number and small AS number. The analysis of the incon-

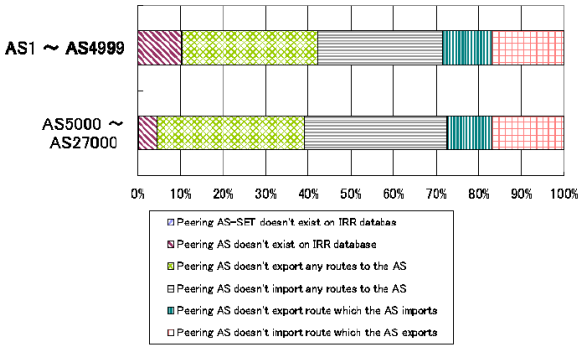

Figure 15: The specification of the inconsistency

sistency of the each group is shown in Figure 15. Figure 15 shows that the group with small AS number has higher rates of *Peering AS doesn't exist on IRR database* than the group with large AS number.

In other words, AS especially with small AS number is required to check whether the peer exists in the database. Irrespective of the AS number, operators need to update their policies frequently.

The detail of the inconsistency in all of import and export sentences are shown in Table 2. In Table 2, "Rate" column shows the rate of each inconsistencies in all 194,820 *import* and *export* sentences. In Table 2, *Peer AS doesn't export any routes to the AS* and *Peer AS doesn't import any routes to the AS* occupy 20% of all the import and export sentences.

In other words, although the peer AS exists in the IRR database, most of them don't specify the AS as peer. RPSL is designed to describe the routing configuration particularly by import and export sentences. Operators can increase their efficiency of operation on BGP network. However, we found out that this functionality is scarcely working.

7 Related Work

Routing Registry Consistency Check (RRCC) project [9] reports inconsistency between IRR database and the real Internet. Tools which detect inconsistencies are available on their web. The inconsistency which they mean is: route prefix which isn't advertised on the real Internet although it is registered in the IRR database, and the route prefix which isn't registered in the IRR database although it is advertised on the real Internet. On the other hand, this research detects the inconsistency among the registered policies. Both of these research aim to improve integrity of the IRR database by correcting the inconsistency of the policies. Therefore, these researches complement each other.

8 Conclusion and Future Work

A mechanism for preventing the increase of the inconsistency of IRR database record has been presented, which we call Policy Check Server. Policy Check Server consists of two components that are *Policy Checker* and *Database Checker*.

We defined this inconsistency as following two categories: inconsistency in routing information import, and routing information export. Both of them can potentially disrupt the connectivity between peering ASes.

Based on this classification, we proposed Policy Check Server. Policy Checker gives the operator an opportunity to inspect the policy which he intends to register to IRR database. Database Checker is a system to investigate the consistency of AS policies in the whole IRR databases in the world.

As a result of the investigation by Database Checker, we have found out 55.8% of ASes has at least one inconsistency. We advocate that the operator of AS should take the consistency between other ASes' policies into consideration before he registers his AS's policy to IRR.

In near future, we intend to apply Policy Check Server to JPIRR which is an IRR server maintained by JPNIC and provide a service to inspect the consistency.

Acknowledgment

The authors would like to thank members of JPNIC IRR Planning Team for many constructive discussions.

REFERENCES

[1] Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). http://www.ietf.org/rfc/rfc1771.txt, March 1995.

[2] C. Huitema. *Routing in the Internet*. SHOEISHA, 2000. (in Japanese).

[3] S. A. Misel. Wow, as7007!. nanong mail archives. http://www.merit.edu/mail.archives /nanog/1997-04/msg00340.html, April 2001.

[4] J. Farrar. C&w routing instability. nanog mail archives. http://www.merit.edu/mail.archives /nanog/2001-04/msg00209.html, April 2001.

[5] R. Mahajan, D. Wetherall, and T. Anderson. Understanding bgp misconfiguration. *SIGCOMM'02*, August 2002.

[6] T. G. Griffin and G. Wilfong. On the correctness of ibgp configuration. *SIGCOMM'02*, August 2002.

[7] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpastra. Routing policy specification language (rpsl). http://www.ietf.org/rfc/rfc2622.txt, June 1999.

[8] Y. Hori, Z. Ikenaga, Y. Kadobayashi, and S. Goto. *Interconnection of Networks*. Iwanami Shoten Publishers, 2001. (in Japanese).

[9] E. Gunduz, S. Kerr, A. Robachevsky, and J. L. S. Damas.

Table 2: Detail of inconsistencies

	Classification	Number of inconsistencies	Rate
1	Peering AS-SET doesn't exist on IRR database	482	0.2%
2	Peering AS doesn't exist on IRR database	7,971	4.0%
3	Peering AS doesn't export any routes to the AS	36,333	18.6%
4	Peering AS doesn't import any routes from the AS	34,710	17.8%
5	Peering AS doesn't export route which the AS imports	11,436	5.8%
6	Peering AS doesn't import route which the AS exports	17,753	9.1%
	Total	108,685	55.8%

Routing registry consistency check. Technical report, RIPE NCC, December 2001.