Representation of IP Routing Policies in the RIPE Database

Tony Bates
Jean-Michel Jouanigot
Daniel Karrenberg
Peter Lothberg
Marten Terpstra

ripe-81

February, 1993

Table of Contents

1. Organisation of this Paper

This paper acts as both a basic tutorial for understanding routing policy and a proposal for additional objects within the RIPE database to store routing policies. Section 2 details some of the history and motivations behind this paper. Section 3 details the general requirements sought for such a method of representation of IP routing policies. Section 4 examines the main concepts, how routing policies are viewed and introduces the terms used throughout this document. Section 5 details the proposed database objects themselves. Section 6 gives a fairly extensive "walk through" of how these objects are used for expressing routing policy and the general principles behind their use. Section 7 explains how these objects should be managed and updated in terms of the RIPE database itself. A short summary is given in section 8. A glossary is also included for clarification and reference. Appendix A and B document the formal syntax for the database objects proposed. The remaining appendices are essentially "for further study" items which may be included or are felt useful to the reader to understand how the representation concept will work in practice.

2. Background

RIPE is a collaborative body of European Regional IP network operators and service providers. European IP networking through its natural evolvement has grown into an extremely complex infrastructure and topology which in essence has even today no real single ‘coordinated core’. Consequently, there are many differing routing policies used within Europe. The RIPE Routing Working Group was formed to tackle the issues involved in large scale routing collaboration as is needed within Europe. To achieve the needed global connectivity, the network operators and service providers need a way to easily and clearly express the various routing policies and provide a method for problem diagnosis and trouble shooting when problems occur.

A RIPE recommendation paper known as ‘ripe-60’, titled "Policy based routing within RIPE", was the outcome of a large amount of effort and discussion within the RIPE Routing Working Group. This tackled many of the issues relevant to the representation of IP routing policies and policy based routing within the RIPE community. It cites many of the pertinent documents addressing the issues of policy based routing and proposes a method for representing routing policy within the RIPE database. At the time of writing of ‘ripe-60’, it was very unclear as to what direction the European IP infrastructure and routing implementations would take and was very much focused in terms of generality.

It has become apparent that some of the problems with ‘ripe-60’ lie in its generalised approach and perhaps somewhat abstract notation that in practice are not easy for network operators to understand. Consequently, the authors of this document feel some clarification is needed and also some enhancement in terms of the current update in routing technology. A conscious effort has been made to be somewhat less general than ‘ripe-60’ in order to improve applicability and clarity.

We would like to express our thanks to Antonio Blasco Bonitio, Piet Beertema, Andrew Partan and Brian Schifflet for their valuable input and comments.

3. Requirements

This section describes the requirements of a scheme for representing routing information in the RIPE database.

Clarity

The representation must be easily explainable to network operators. If it is not, then it will not be used or worse, will be used incorrectly. This also includes that it must either map well to the way network operators think about routing policy or provide them with a convenient way to start thinking about it.

Translatability

The representation must be translatable to and from router configuration files. The resulting configurations must be efficient in the sense that they are implementable with current router resources even in complex environments. If this requirement is not met there will be too many inconsistencies between the published and the implemented policies which will in turn cause hard to detect routing problems.

Checkability

It must be possible to check the consistency of the published policies. Thus some level of redundancy is welcome if it adds value by making it possible to reveal inconsistencies.

Applicability

It must be possible to easily build diagnostic tools which help network operators to diagnose problems by making use of the policy information. These tools should answer questions like:

- "Which paths between network A and B are allowed by published policies ?"
- "Which path will traffic between network A and network B take when a certain routing path cannot be used for transit ?"
- "Does network A somehow have connectivity to all other networks within RIPE ?"

Generality

The "Clarity" and "Translatability" requirements may push the design into the direction of large aggregations of networks by routing and administrative domains. While this serves those requirements well the representation must make it possible to specify particular policies down to network granularity. Specifically, it should be possible to specify forwarding restrictions.

4. General Representation of Policy Information

Networks, Network Operators and Autonomous Systems

Throughout this document an effort is made to be consistent with terms so as not to confuse the reader.

When we talk about "networks" we mean physical networks which have a unique IP network number: Layer 3 entities. We do not mean organisations.

We call the organisations operating networks "network operators". For the sake of the examples we divide network operators into two categories: "service providers" and "customers". A "service provider" is a network operator who operates a network to provide Internet services to different organisations, its "customers". The distinction between service providers and customers is not clear cut. A national research networking organisation frequently acts as a service provider to Universities and other academic organisations, but in most cases it buys international connectivity from another service provider. A University networking department is a customer of the research networking organisation but in turn may regard University departments as its customers.

An Autonomous System (AS) is a group of IP networks having a single clearly defined routing policy which is run by one or more network operators. Inside ASes IP packets are routed using one or more Interior Routing Protocols (IGP). In most cases interior routing decisions are based on metrics derived from technical parameters like topology, link speeds and load (1).

ASes exchange routing information with other ASes using Exterior Routing Protocols (EGP). Exterior routing decisions are frequently based on policy based rules rather than purely on technical parameters. Until now, there are no tools to configure complex policies and to communicate those policies between ASes while still ensuring proper operation of the Internet as a whole. Some EGPs like BGP-3 and BGP-4 provide tools to filter routing information according to policy rules and more. None of them provides a mechanism to publish or communicate the policies themselves. Yet this is critical for operational coordination and fault isolation among network operators and thus for the operation of the global Internet as a whole.

This paper proposes to use the RIPE database as a repository for publishing the policies of European ASes. There are two types of policy information which need to be represented in the database. These are "routing policies" and "access policies".

Routing Policies

The exchange of routing information between ASes is subject to routing policies. Consider the case of two ASes, X and Y exchanging routing information:

```
NET1 ......  ASX  <--->  ASY  ....... NET2
```

ASX knows how to reach a network called NET1. It does not matter whether NET1 is belonging

(1) The entity we refer to as an AS is frequently and more generally called a routing domain with the AS just being an implementation vehicle. We have decided to use the term AS exclusively because it relates more directly with the database objects and routing tools. By using only one term we hope to reduce the number of concepts and to avoid confusion. The academically inclined reader may forgive us.

to ASX or some other AS which exchanges routing information with ASX either directly or indirectly; we just assume that ASX knows how to direct packets towards NET1. Likewise ASY knows how to reach NET2.

In order for traffic from NET2 to NET1 to flow between ASX and ASY, ASX has to announce NET1 to ASY using an external routing protocol. This states that ASX is willing to accept traffic directed to NET1 from ASY. Policy thus comes into play first in the decision of ASX to announce NET1 to ASY.

In addition ASY has to accept this routing information and use it. It is ASY's privilege to either use or disregard the information that ASX is willing to accept traffic for NET1. ASY might decide not to use this information if it does not want to send traffic to NET1 at all or if it considers another route more appropriate to reach NET1.

So in order for traffic in the direction of NET1 to flow between ASX and ASY, ASX must announce it to ASY and ASY must accept it from ASX:

```
  resulting packet flow towards NET1
<<===================================
                  |
                  |
   announce NET1  |  accept NET1
  --------------> + ------------->
                  |
      AS X        |    AS Y
                  |
   <------------- + <--------------
     accept NET2  |  announce NET2
                  |
                  |
 resulting packet flow towards NET2
 ===================================>>
```

Ideally, and seldom practically, the announcement and acceptance policies of ASX and ASY are identical.

In order for traffic towards NET2 to flow, announcement and acceptance of NET2 must be in place the other way round. For almost all applications connectivity in just one direction is not useful at all.

It is important to realise that with current destination based forwarding technology routing policies must eventually be expressed in these terms. It is relatively easy to formulate reasonable policies in very general terms which CANNOT be expressed in terms of announcing and accepting networks. With current technology such policies are almost always impossible to implement. See appendix E for further explanation of routing policy limitations.

Usually separate policies are not configured for each network separately but for groups of networks. In practise these groups are almost always defined by the networks forming one or more ASes.

By expressing routing policy in this way it should be noted that this information is only of use when you have full information for all networks within the global Internet. Equally, the current detail is only limited to the RIPE database and for an AS exchanging routing reachability information for an AS whose information is not in the database we obviously cannot give a full representation of the ASes routing policy.

Access Policies

Access policies contrary to routing policies are not necessarily defined in terms of ASes. The very simplest type of access policy is to block packets from a specific network S from being forwarded to another network D. A common example is when some unappropriate use of resources on network D has been made from network S and the problem has not been resolved yet. Other examples of access policies might be resources only accessible to networks belonging to a particular disciplinary group or community of interest. While most of these policies are better implemented at the host or application level, network level access policies do exist and are a source of connectivity problems which are sometimes hard to diagnose. Therefore they should also be documented in the database according to similar requirements as outlined above.

Tools

The network operators will need a series of tools for policy routing. Some tools are already available to perform some of the tasks. Most notably, the "Network List Compiler", NLC produced as part of earlier work done by the RIPE Routing Working Group. These tools will enable them to use the routing policy stored in the RIPE database to perform such tasks as check actual routing against policies defined, ensure consistency of policies set by different operators, and simulate the effects of policy changes.

The tools are expected to include:

"prcheck"

A tool to check the consistency of routing policies stored in the database. This tool will flag if two neighboring network operators specify conflicting or inconsistent routing information exchanges with each other and also detect global inconsistencies where possible.

"prpath"

Extract all AS-paths between two networks which are allowed by routing policy from the routing policy database.

"prconn"

Display the connectivity a given network has according to current policies. This will of course also be able to find the set of networks a given network can not reach.

"prtraceroute"

A version of the existing traceroute tool which will be able to display whether a route in use is allowed by policy and where deviations from policy occur.

5. Proposed Database Objects

Autonomous System (AS)

An Autonomous System (AS) is a group of IP networks run by one or more network operators which has a single and clearly defined routing policy.

An AS has a unique number associated with it which is used in both exchange of exterior routing information (i.e. network reachability information between ASes) and as an identifier of the AS itself. Exterior routing protocols such as BGP and EGP are used to exchange routing information between ASes.

In routing terms an AS will normally use one or more interior gateway protocols in conjunction with some sort of common agreed metrics when exchanging network information within its own AS.

The term AS is often confused or even misused as a convenient way of grouping together a set of networks which belong under the same administrative umbrella even if within that group of networks there are various different routing policies. We provide the "community" concept for such use. ASes can strictly have only one single routing policy.

The creation of an AS should be done in a conscious and well coordinated manner to avoid creating ASes for the sake of it, perhaps resulting in the worst case scenario of one AS per IP network number. It should be noted that there is a limited number of AS numbers available. This may mean that by applying the general rules for the creation and allocation of an AS below, some re-engineering may well be needed. However, this may be the only way to actually implement the desired routing policy anyway. The creation and allocation of an AS should be done with the following recommendations in mind:

- Creation of an AS is only required when exchanging routing information with other ASes. Some router implementations make use of an AS number as a form of tagging to identify the routing process. However, it should be noted that this tag does not need to be unique unless routing information is indeed exchanged with other ASes.

- An IP network number can and must only belong to one AS. This is a direct consequence of the fact that at each point in the Internet there can only be exactly one routing policy for traffic destined to each network. In the case of the IP network which is used in neighbor peering between two ASes, say at the border between two ASes, a conscious decision must be made as to which AS this IP network number actually resides in.

- For a simple case of customer networks connected to a single service provider, the IP network should be a member of the service provider's AS. In terms of routing policy the IP network has exactly the same policy as the service provider and there is no need to make any distinction in routing information. This idea may at first seem slightly alien to some, but it highlights the clear distinction in the use of the AS number as a representation of routing policy as opposed to some form of administrative use.

- If a network operator connects to more than one AS with different routing policies then they need to create their own AS. In the case of multi-homed customer networks connected to two service providers there are at least two different routing policies to a given customer network. At this point the customer networks will be part of a single AS and this AS would be distinct from either of the service providers ASes. This allows the customer the ability of having a different representation of policy and preference to the different service providers. This is the ONLY case where a network operator should create its own AS number.

- As a general rule one should always try to populate the AS with as many IP networks as possible, providing all IP networks conform to the same routing policy.

With the above rules in mind, it follows that when a network operator changes service provider, it will move from one AS to another. Once the database is being used as hoped for routing then a clear procedure is needed for making sure any AS transition takes place in a smooth and transparent manner without any loss of connectivity. See Appendix D for detail of how this should be done.

Each AS is represented in the RIPE database by both an AS object and AS tags on the network objects representing the networks belonging to the AS. The AS object stores descriptive, administrative and contact information about the AS as well as the routing policies of the AS in relation to all neighboring ASes.

The AS tags on the network objects define the set of networks belonging to an AS. Each network can have exactly one AS tag. The AS tags can only be created and updated by the "guardian" of the AS and not by those immediately responsible for the particular network. This ensures that operators, especially service providers, remain in control of AS membership. The example below shows how this would be represented in terms of the network object. The tag "aut-sys" is used. Refer to ‘ripe-50’, for a complete description of the network object in the RIPE database.

Example:

```
inetnum:    192.87.45.0
netname:    RIPE-NCC
descr:      RIPE Network Coordination Centre
descr:      Amsterdam, Netherlands
country:    NL
admin-c:    Daniel Karrenberg
tech-c:     Marten Terpstra
connect:    RIPE NSF
aut-sys:    1104
rev-srv:    ns.ripe.net
changed:    marten@ripe.net 930121
source:     RIPE
```

The AS object itself is used to represent a description of administrative details and the routing policies of the AS itself. The AS object definition is depicted as follows.

Example:

```
aut-num:  AS1104
descr:    NIKHEF-H Autonomous system
as-in:    AS1213 100 AS1213
as-in:    AS1913 100 AS1913
as-in:    AS1755 150 ANY
as-out:   AS1213 ANY
as-out:   AS1913 ANY
as-out:   AS1755 AS1104 AS1913 AS1213
tech-c:   Rob Blokzijl
admin-c:  Eric Wassenaar
guardian: as-guardian@nikhef.nl
source:   RIPE
changed:  k13@nikhef.nl 920713
changed:  ripe-dbm@ripe.net 920910
```

See Appendix A for a complete syntax definition of the "aut-num" object.

It should be noted that this representation provides two things:

- a set of networks.
- a description of administrative details and routing policies.

The set of networks can be used to generate network list based configuration information as well as configuration information for exterior routing protocols knowing about ASes. This means an AS can be defined and is useful even if it does not use routing protocols which know about the AS concept!

Community

A community is a group of networks that cannot be represented by an AS or a group of ASes. It is in some circumstances useful to define a group of networks that have something in common. This could be a special access policy to a supercomputer centre, a group of networks used for a specific mission, or a disciplinary group that is scattered among several autonomous systems. Also these communities could be useful to group networks for the purpose of network statistics.

Communities do not exchange routing information, since they do not represent an autonomous system. More specifically, communities do not define routing policies, but access or usage policies. Contrary to an AS, networks can belong to more than one community.

Communities should be defined in a strict manner, to avoid creating as many communities as there are networks, or even worse. Communities should be defined following the two rules below;

- Communities must have a global meaning. Communities that have no global meaning, are used only in a local environment and must be avoided.

- Communities must not be defined to express non-local policies. It should be avoided that a community is created because some other organisation forces a policy upon your organisation. Communities must only be defined to express a policy defined by your organisation.

Community examples

There are some clear examples of communities:

PROVIDER —
all customers of a given service provider even though they can have various different routing policies and hence belong to different ASes. This would be extremely useful for statistics collection.

HEPnet —
the High Energy Physics community partly shares infrastructure with other organisations, and the institutes it consists of are scattered all over Europe, often being part of a non HEPnet autonomous system. To allow statistics or access policies, a community HEPnet, consisting of all networks that are part of HEPnet, conveniently groups all these networks.

NSFNET —
the National Science Foundation Network imposes an acceptable use policy on networks that wish to make use of it. A community NSFNET could imply the set of networks that comply to this policy.

MULTI —
a large multinational corporation that does not have its own internal infrastructure, but connects to the various parts of its organisations by using local service providers that connect them all together, may decide to define a community to restrict access to their networks, only by networks that are part of this community. This way a corporate network could be defined on shared infrastructure. Also, this community could be used by any of the service providers to do statistics for the whole of the corporation, for instance to do topology or bandwidth planning.

Similar to Autonomous systems, each community is represented in the RIPE database by both a community object and community tags on the network objects representing the networks belonging to the community. The community object stores descriptive, administrative and contact information about the community.

The community tags on the network objects define the set of networks belonging to a community. A network can have multiple community tags. The community tags can only be created and updated by the "guardian" of the community and not by those directly responsible for the particular network. This ensures that guardians remain in control of community membership.

Here's an example of how this might be represented in terms of the community tags within the network object. We have an example where the network 192.16.199.0 has a single routing policy (i.e. that of AS 1104), but is part of several different communities of interest. We use the tag "comm-list" to represent the list of communities associated with this network. NIKHEF-H uses the service provider SURFnet (a service provider with customers with more than one routing policy), is also part of the High Energy Physics community as well as having the ability to access the

Supercomputer at CERN (2).

Example:

```
inetnum:   192.16.199.0
netname:   HEFNET
descr:     Local Ethernet
descr:     NIKHEF section H
descr:     Science Park Watergraafsmeer
descr:     Amsterdam
descr:     The Netherlands
country:   NL
admin-c:   Eric Wassenaar
tech-c:    Rob Blokzijl
aut-sys:   1104
comm-list: HEPNET CERN-SUPER SURFNET
gateway:   nih
changed:   ripe-dbm@ripe.net 920604
source:    RIPE
```

In the above examples some communities have been defined. The community object itself will take the following format:

Example:

```
community:  SURFnet
descr:      Dutch academic research network
authority:  SURFnet B.V.
guardian:   comm-guardian@surfnet.nl
admin-c:    Erik-Jan Bos
tech-c:     Erik-Jan Bos
changed:    ripe-dbm@ripe.net 920604
source:     RIPE
```

For a complete explanation of the syntax please refer to Appendix B.

(2) The community ‘CERN-SUPER’, is somewhat notional, but is intended as an example of a possible use of an access policy constraint.

6. Representation of Routing Policies

Routing policies of an AS are represented in the autonomous system object. Initially we show some examples, so the reader is familiar with the concept of how routing information is represented, used and derived. Refer to Appendix A, for the full syntax of the "aut-num" object.

The topology of routing exchanges is represented by listing how routing information is exchanged with each neighboring AS. This is done separately for both incoming and outgoing routing information. In order to provide backup and back door paths a relative cost is associated with incoming routing information.

Example 1:

```
AS1------AS2
```

This specifies a simple routing exchange of two presumably isolated ASes. Even if either of them has routing information about networks in ASes other than AS1 and AS2, none of that will be announced to the other.

```
aut-num:   AS1
as-out:    AS2 AS1      # announce all networks in AS1 to AS2
as-in:     AS2 100 AS2  # accept all AS2 networks from AS2

aut-num:   AS2
as-out:    AS1 AS2      # announce all networks in AS2 to AS1
as-in:     AS1 100 AS1  # accept all AS1 networks from AS1
```

The number 100 in the inbound specifications is a relative cost, which is used for backup and back door routes. The absolute value is of no significance. The relation between different values within the same AS object is. A lower value means a lower cost. This is consciously similar to the cost based preference scheme used with DNS MX RRs.

Example 2:

Now suppose that AS2 is connected to one more AS, besides AS1, and let’s call that AS3:

```
AS1------AS2------AS3
```

In this case there are two reasonable routing policies:

a) AS2 just wants to exchange traffic with both AS1 and AS3 itself without passing traffic between AS1 and AS3.

b) AS2 is willing to pass traffic between AS3 and AS1, thus acting as a transit AS

Example 2a:

In the first case AS1's representation in the database will remain unchanged as will be the part of AS2's representation describing the routing exchange with AS1. A description of the additional routing exchange with AS3 will be added to AS2's representation:

```
aut-num:   AS1
as-out:    AS2 AS1      # announce all networks in AS1 to AS2
as-in:     AS2 100 AS2  # accept all AS2 networks from AS2

aut-num:   AS2
as-out:    AS1 AS2      # announce only networks in AS2 to AS1
as-in:     AS1 100 AS1  # accept all AS1 networks from AS1
as-out:    AS3 AS2      # announce only networks in AS2 to AS3
as-in:     AS3 100 AS3  # accept all AS3 networks from AS3

aut-num:   AS3
as-out:    AS2 AS3      # announce all networks in AS3 to AS2
as-in:     AS2 100 AS2  # accept all AS2 networks from AS2
```

Note that in this example, AS2 keeps full control over its resources. Even if AS3 and AS1 were to allow each other's routes in from AS2, the routing information would not flow because AS2 is not announcing it (3).

(3) Of course AS1 and AS3 could just send traffic to each other to AS2 even without AS2 announcing the networks, hoping that AS2 will forward it correctly. Such questionable practises however are beyond the scope of this document.

Example 2b:

If contrary to the previous case, AS1 and AS3 are supposed to have connectivity to each other via AS2, all AS objects have to change:

```
aut-num:   AS1
as-out:    AS2 AS1          # announce all networks in AS1 to AS2
as-in:     AS2 100 AS2 AS3  # accept AS2 and AS3 networks from AS2

aut-num:   AS2
as-out:    AS1 AS2 AS3      # announce all networks in AS2 and AS3 to AS1
as-in:     AS1 100 AS1      # accept all AS1 networks from AS1
as-out:    AS3 AS2 AS1
as-in:     AS3 100 AS3

aut-num:   AS3
as-out:    AS2 AS3          # announce all networks in AS3 to AS2
as-in:     AS2 100 AS1 AS2  # accept AS2 and AS1 networks from AS2
```

Note that the amount of routing information exchanged with a neighbor AS is defined in terms of networks belonging to ASes. In BGP terms this is the AS where the routing information originates and the originating AS information carried in BGP could be used to implement the desired policy. However, using BGP or the BGP AS-path information is not required to implement the policies thus specified. Configurations based on network lists can easily be generated from the database. The AS path information, provided by BGP can then be used as an additional checking tool as desired.

The specification understands one special expression and this can be expressed as a boolean expressions:

ANY -

means any routing information known. For output this means that all networks an AS knows about are announced. For input it means that anything is accepted from the neighbor AS.

Example 3:

AS4 is a stub customer AS, which only talks to service provider AS123.

```
                |
                |
           -----AS123------AS4
                |
                |
```

```
aut-num: AS4                # representation of AS4 in the database
as-out:  AS123 AS4          # announce only networks in AS4 to AS123
as-in:   AS123 100 ANY      # accept any announcement from AS123

aut-num: AS123              # representation of the service provider AS123
as-in:   AS4 100 AS4        # accept routes for customer AS4's networks
as-out:  AS4 ANY            # provide all routes to customer AS4
<further neighbors>
```

Since AS4 has no other way to reach the outside world than AS123 it is not strictly necessary for AS123 to send routing information to AS4. AS4 can simply send all traffic for which it has no explicit routing information to AS123 by default. This strategy is called default routing. It is expressed in the database by adding one or more default tags to the autonomous system which uses this strategy. In the example above this would look like:

```
aut-num: AS4                # representation of AS4 in the database
as-out:  AS123 AS4          # announce only networks in AS4 to AS123
default: AS123 100          # send default traffic to AS123

aut-num: AS123              # representation of the service provider AS123
as-in:   AS4 100 AS4        # accept routes for customer AS4's networks
                            # no announcements to AS4, AS4 defaults to us
<further neighbors>
```

Example 4:

AS4 now connects to a different operator, AS5. AS5 uses AS123 for outside connectivity but has itself no direct connection to AS123. AS5 traffic to and from AS123 thus has to pass AS4. AS4 agrees to act as a transit AS for this traffic.

```
                         |
                         |
                  -----AS123------AS4-------AS5
                         |
                         |
```

```
aut-num:    AS4             # representation of AS4 in the database
as-out:     AS123 AS4 AS5   # own and AS5 networks to AS123
as-in:      AS123 100 ANY   # accept any announcement from AS123
as-out:     AS5 ANY         # send all announcements through to AS5
as-in:      AS5 50 AS5      # accept only AS5 announcements from them

aut-num:    AS5
as-in:      AS4 100 ANY
as-out:     AS4 AS5

aut-num:    AS123           # representation of the service provider AS123
as-in:      AS4 100 AS4 AS5 # accept AS4's own and AS5 from AS4
as-out:     AS4 ANY         # all to customer AS4 and via it to AS5
<further neighbors>
```

Now AS4 has two sources of external routing information. AS5 which provides only information about its own networks and AS123 which provides information about the external world. Note that AS4 accepts information about AS5 from both AS123 and AS5 although AS5 information cannot come from AS123 since AS5 is connected only via AS4 itself. The lower cost of 50 for the announcement from AS5 itself compared to 100 from AS123 ensures that AS5 is still believed even in case AS123 will unexpectedly announce AS5.

In this example too, default routing can be used by AS5 much like in the previous example. AS4 can also use default routing towards AS123:

```
aut-num:    AS4             # representation of AS4 in the database
as-out:     AS123 AS4 AS5   # own and AS5 networks to AS123
default:    AS123 11        # default to AS123
as-in:      AS5 50 AS5      # accept only AS5 announcements from them
                            # no announcements to AS5, they default to us

aut-num:    AS5
as-out:     AS4 AS5
default:    AS4 100

aut-num:    AS123           # representation of the service provider AS123
as-in:      AS4 100 AS4 AS5 # accept AS4's own and AS5 from AS4
                            # no announcements to AS4, they default to us
<further neighbors>
```

Note that the relative cost associated with default routing is totally separate from the relative cost associated with inbound announcements. The default route will never be taken if an explicit route is known to the destination. Thus an explicit route can never have a higher cost than the default route. The relative cost associated with the default route is only useful in those cases where one wants to configure multiple default routes for redundancy.

Note also that in this example the configuration using default routes has a subtly different behaviour than the one with explicit routes: In case the AS4-AS5 link fails AS4 will send traffic to AS5 to AS123 when using the default configuration. Normally this makes not much difference as there will be no answer and thus little traffic. With certain datagram applications which do not require acknowledgements however, significant amounts of traffic may be uselessly directed at AS123. Similarly default routing should not be used if there are stringent security policies which proscribe any traffic intended for AS5 to ever touch AS123.

Generally it can be said that default routing should only be used in very simple topologies. Once the situation gets more complex using default routes can lead to unexpected results or even defeat the routing policies established when links fail. As an example consider how Example 5a) below could be implemented using default routing.

Example 5:

In a different example AS4 has a private connection to AS6 which in turn is connected to the service provider AS123:

```
           |
           |
     -----AS123------AS4
           |          |
           |          |
           |          |
          AS6 ---------+
```

There are a number of policies worth examining in this case:

a) AS4 and AS6 wish to exchange traffic between themselves exclusively via the private link between themselves; such traffic should never pass through the backbone (AS123). The link should never be used for transit traffic, i.e. traffic not both originating in and destined for AS4 and AS6.

b) AS4 and AS6 wish to exchange traffic between themselves via the private link between themselves. Should the link fail, traffic between AS4 and AS6 should be routed via AS123. The link should never be used for transit traffic.

c) AS4 and AS6 wish to exchange traffic between themselves via the private link between themselves. Should the link fail, traffic between AS4 and AS6 should be routed via AS123. Should the connection between AS4 and AS123 fail, traffic from AS4 to destinations behind AS123 can pass through the private link and AS6's connection to AS123.

d) AS4 and AS6 wish to exchange traffic between themselves via the private link between themselves. Should the link fail, traffic between AS4 and AS6 should be routed via AS123. Should the backbone connection of either AS4 or AS6 fail, the traffic of the disconnected AS should flow via the other AS's backbone connection.

Example 5a:

```
aut-num:   AS4
as-in:     AS123 100 NOT AS6        # accept anything besides AS6
as-out:    AS123 AS4                # announce only ourselves to AS123
as-in:     AS6 50 AS6               # accept only AS6 from AS6
as-out:    AS6 AS4                  # announce only own nets to AS6

aut-num:   AS123
as-in:     AS4 100 AS4
as-out:    AS4 ANY
as-in:     AS6 100 AS6
as-out:    AS6 ANY
<further neighbors>

aut-num:    AS6
as-in:      AS123 100 NOT AS4
as-out:     AS123 AS6
as-in:      AS4 50 AS4
as-out:     AS4 AS6
```

Note that here the configuration is slightly inconsistent. AS123 will announce AS6 to AS4 and AS4 to AS6. These announcements will be filtered out on the receiving end. This will implement the desired policy. Consistency checking tools might flag these cases however.

Example 5b:

```
aut-num:   AS4
as-in:     AS123 100 ANY          # accept anything from AS123
as-out:    AS123 AS4              # announce only ourselves to AS123
as-in:     AS6 50 AS6             # accept AS6 only from AS6
as-out:    AS6 AS4                # announce only own nets to AS6

aut-num:   AS123
as-in:     AS4 100 AS4
as-out:    AS4 ANY
as-in:     AS6 100 AS6
as-out:    AS6 ANY
<further neighbors>

aut-num:   AS6
as-in:     AS123 100 ANY
as-out:    AS123 AS6
as-in:     AS4 50 AS4
as-out:    AS4 AS6
```

The thing to note here is that in the ideal operational case, ‘all links working’ AS4 will receive announcements for AS6 from both AS123 and AS6 itself. In this case the announcement from AS6 will be preferred because of its lower cost and thus the private link will be used as desired. AS6 is configured as a mirror image.

Example 5c:

The new feature here is that should the connection between AS4 and AS123 fail, traffic from AS4 to destinations behind AS123 can pass through the private link and AS6's connection to AS123.

```
aut-num:  AS4
as-in:    AS123 100 ANY     # expect anything from AS123
as-out:   AS123 AS4         # announce only ourselves to AS123
as-in:    AS6 50 AS6        # accept AS6 only from AS6
as-in:    AS6 110 ANY       # accept anything from AS6 (worse than AS123)
as-out:   AS6 AS4           # announce only own nets to AS6

aut-num:  AS123
as-in:    AS4 1 AS4
as-out:   AS4 ANY
as-in:    AS6 1 AS6
as-in:    AS6 2 AS4         # now also accept AS4 from AS6 in case backup
as-out:   AS6 ANY
<further neighbors>

aut-num:    AS6
as-in:      AS123 100 ANY
as-out:     AS123 AS6 AS4   # for backup also pass AS4 to AS123
as-in:      AS4 50 AS4
as-out:     AS4 ANY         # for backup export all we know to AS4
```

Note that it is important to make sure to propagate routing information for both directions in backup situations like this. Connectivity in just one direction is not useful at all for almost all applications.

Note also that in case the AS6-AS123 connection breaks, AS6 will only be able to talk to AS4. The symmetrical case (5d) is left as an exercise to the reader.

7. RIPE Database Update Procedure

This section, describes a procedure for updating the routing attributes in the RIPE database as mentioned in sections 5 and 6. These routing tags are essential for the routing of networks that are known in the RIPE database. Therefore updates of these attributes must be:

- Properly authorised
- Guarded against typing errors
- Efficient for both maintainers of the attributes and the maintainers of the whole database

The procedure

For each of the AS and Community tags, a list of all networks having this attribute is kept separately from the general database itself. These lists will be the ‘only’ source of AS and community membership information used in the database. Normal database updates ‘never’ change these attributes. If an update includes such an attribute and a discrepancy between the values in the update and those in the database is found, a diagnostic message will be sent to the originator of the update. The attributes as defined in the files are incorporated in the database at each normal update run. To ensure proper control and authorisation, we propose to maintain the lists on the central NCC database server, where each of the guardians of a tag keeps track of his or her own list. The lists are maintained through individual logins for each of the guardians. The guardians can themselves decide in what manner they want to update their list. The NCC will offer interactive logins, ftp logins or any other means that might be deemed useful.

File Format

We propose to keep the file format as simple as possible. The name of the file should indicate the name of the attributes. This means that there cannot be two attributes with the same name. The format within the file we propose as the "inetnum:" entry for each of the networks, separated by an empty line. If a guardian feels that he would want more fields to identify a network that will get his attribute, he is free to do so. An attribute will only be added to a network if all fields defined in the list match a network in the database.

Advantages

The update procedure as proposed above has the following advantages:

- Authorisation of adding/deleting is guaranteed.
- No need for mailing back and forth of authorisation messages.
- Simple procedure for both database maintainers and guardians.
- Guardians keep full control of their attribute.
- Could be implemented at the NCC without too much delay.

8. Summary

In summary this paper basically introduces two new RIPE database objects, the autonomous system — "aut-num" and the community — "community" object. It attempts to clarify terms in such a way as to make them easier and more usable for network operators. It provides a mechanism to easily and clearly express routing policy and provide a mechanism to create, maintain and administer network based filter lists for routing policy. Through its combination of tutorial and outlined proposal style we have hopefully managed to satisfy the requirements as set out in Section 3. The paper also keeps in mind the original need for generalisation whilst at the same time catering somewhat for the growing deployment of newer more advanced routing protocols and their use.

9. Glossary

It is felt useful to include a basic glossary of some of the terms used within this paper to act as both a reference and as possible clarification tool.

— **Announce**

In terms of this paper, to announce means to convey reachability information for a given IP network. This can be done using a variety of routing techniques. For example, by use of redistribution of routes into a dynamic routing protocol. Equally this can be done by installing a static route which effectively announces the route to ones own router in the routing context. This then can also re-distributed if so desired.

— **Autonomous System (AS)**

A group of IP networks run by one or more network operators having a single clearly defined routing policy.

— **BGP-3/BGP-4**

BGP is the Border Gateway Protocol, BGP-3 is the current Internet standard with BGP-4 as the proposed standard. This is an exterior based IP routing protocol with some interior based routing properties. It has the ability to carry AS-Path information along with routing reachability information. For further information, refer to the following papers;

BGP-3 • "A Border Gateway Protocol 3 (BGP-3)", RFC 1276. K. Lougheed, Y. Rekhter, October 1991.

BGP-4 • "A Border Gateway Protocol 4 (BGP-4)", Work in Progress. Y. Rekhter, T.Li, December 1992.

— **Network**

A layer 3 entity, not any form of organisation. For all intent and purposes, the physical network represented by a unique IP network number.

— **Network Operator**

An organisation which operates networks. A network operator can be a customer of another service provider as well as being a service provider to customers itself.

— **RIPE database**

A European based database containing information on IP networks, persons, namespace administration data and routing policies.

— **Service Provider**

A network operator who provides Internet services to different organisations.

10. Author’s Addresses

Tony Bates
RIPE Network Coordination Centre
Kruislaan 409
NL-1098 SJ Amsterdam
The Netherlands
+31 20 592 5065
T.Bates@ripe.net

Jean-Michel Jouanigot
CERN, European Laboratory for Particle Physics
CN division
CH-1211 Geneva 23
Switzerland
+41 22 767 4417
jimi@dxcoms.cern.ch

Daniel Karrenberg
RIPE Network Coordination Centre
Kruislaan 409
NL-1098 SJ Amsterdam
The Netherlands
+31 20 592 5065
D.Karrenberg@ripe.net

Peter Lothberg
STUPI
Box 9129
102 72 Stockholm
Sweden
+46 8 6699720
roll@stupi.se

Marten Terpstra
RIPE Network Coordination Centre
Kruislaan 409
NL-1098 SJ Amsterdam
The Netherlands
+31 20 592 5065
M.Terpstra@ripe.net

Appendix A — Syntax for the "aut-num" object.

Here is a summary of the tags associated with aut-num object itself:

```
aut-num:       [mandatory]
descr:         [mandatory]          [multiple]
as-in:         [optional]           [multiple]
as-out:        [optional]           [multiple]
default:       [optional]           [multiple]
tech-c:        [mandatory]          [multiple]
admin-c:       [mandatory]          [multiple]
guardian:      [mandatory]
remarks:       [optional]           [multiple]
changed:       [mandatory]          [multiple]
source:        [mandatory]          [RIPE]
```

Each item has the following syntax:

aut-num:

Autonomous system number.
This must be a uniquely allocated autonomous system number from an AS registry (i.e. the RIPE NCC, the Inter-NIC, etc).

Format: AS<positive integer between 1 and 65535>
Example:

```
aut-num: AS1104
```

Status: mandatory

descr:

A short description of the Autonomous system.

Format: free text, one line per entry, multiple lines in sequence
Example:

```
descr:    NIKHEF section H
descr:    Science Park Watergraafsmeer
descr:    Amsterdam
```

Stautus: mandatory

as-in:

A description of accepted routing information between AS peers.

Format: <aut-num> <cost> <routing policy expression>,
on multiple lines.

<aut-num> refers to your AS neighbor.

<cost> is a positive integer used to express a relative cost of routes learned. The lower the cost the more preferred the route.

<routing policy expression> can take the following formats.

1. A list of one or more ASes.

Example:
```
as-in: AS1103 100 AS1103
as-in: AS786  105 AS1103
as-in: AS786  10  AS786
as-in: AS1755 110 AS1103 AS786
```

2. A set of KEYWORDS.
The following keywords are currently defined.

ANY — this means anything the neighbor AS knows.
RIPE-DB — any network currently in the RIPE database.
LOCAL — any network in the RIPE database which is part of the community LOCAL (i.e. no connectivity outside its own organisation).

3. A logical expression of either 1 or 2 above
The current logical operators are defined as:

AND
OR
NOT

Rules are grouped together using parenthesis i.e "(" and ")".

Example:
```
as-in: AS1755 100 RIPE-DB AND NOT (LOCAL AS1234 AS513)
as-in: AS1755 150 DEFAULT
```

Stautus: optional

as-out:

A description of generated routing information sent to other AS peers.

Format: <aut-num> <routing policy expression>, multiple lines.

<aut-num> refers to your AS neighbor.

<routing policy expression> is explained in the as-in definition above.

Example:
```
as-out: AS1104 AS978
```

```
as-out: AS1755 ANY
as-out: AS786 RIPE-DB AND NOT (AS978 LOCAL)
```

Stautus: optional

default:

An indication of how default routing is done.

Format: <aut-num> <relative cost>, multiple lines.

<aut-num> The AS peer you will default route to.

<relative cost> The relative cost is a positive integer use to express a preference for default. There is no direct relationship to to the cost used in the "as-in" tag. The lower the cost indicates which AS peer is chosen for default.

Example:

```
default: AS1755 10
default: AS786  5
```

Stautus: optional

tech-c:

Name or NIC-handle of technical contact person.
This is someone to be contacted for technical problems
such as misconfiguration, etc.

Format: <firstname> <initials> <lastname>, multiple lines.

Example:

```
tech-c: John E. Doe
```

Status: mandatory

admin-c:

Name or NIC-handle of administrative contact person. This is probably the name of the guardian but it is not mandatory to be.

Format: <firstname> <initials> <lastname>, multiple lines.

Example:

```
admin-c: Joe T. Bloggs
```

Status: mandatory

guardian:

Malibox of the guardian of the Autonomous system.

Format: <email-address>, single line.

<email-address> This should be in RFC822 format wherever possible.

Example:

guardian: as1104-guardian@nikhef.nl

Status: mandatory

source: RIPE

Source of the information.
This is used to separate information from different sources kept by the same database software.

Format: RIPE

Status: mandatory

changed:

Who and when changed this last.

Format: <email-address> YYMMDD

Example:

changed: johndoe@terabit-labs.nn 900401

Status: mandatory

remark:

Remarks/comments

Format: text string, multiple lines

Example:

```
remark: Multihomed AS talking to AS 1755 and AS786
remark: Will soon connect to AS 1104 also.
```

Status: optional

Appendix B — Syntax details for the "community" object.

Here is a summary of the tags associated with "community" object itself:

```
community:      [mandatory]
descr:          [mandatory]          [multiple]
authority:      [mandatory]
guardian:       [mandatory]
tech-c:         [mandatory]          [multiple]
admin-c:        [mandatory]          [multiple]
remarks:        [optional]           [multiple]
changed:        [mandatory]          [multiple]
source:         [mandatory]          [RIPE]
```

Each item has the following syntax:

community:

Name of community. Should be as representative of the community as possible rather than being something obscure.

Format: Text string which cannot start with "AS" or any of the <routing policy expression> KEYWORDS. See Appendix A.

Example:
community: WCW

Stautus: mandatory

descr:

A short description of the community represented.

Format: free text, one line per entry, multiple lines in sequence
Example:
descr: Science Park Watergraafsmeer
descr: Amsterdam

Stautus: mandatory

authority:

The formal authority for this community. This could be an organisation, institute, etc.

Format: text
Example:
authority: WCW LAN committee

Stautus: mandatory

guardian:

Malibox of the guardian of the community.

Format: <email-address>, single line.

Example:
guardian: wcw-guardian@nikhef.nl

Status: mandatory

tech-c:

Name or NIC-handle of technical contact person.

Format: <firstname> <initials> <lastname>, multiple lines.
Example:
tech-c: John E. Doe

Status: mandatory

admin-c:

Name or NIC-handle of administrative contact person.

Format: <firstname> <initials> <lastname>, multiple lines.
Example:
admin-c: Joe T. Bloggs

Status: mandatory

source: RIPE

Source of the information.
This is used to separate information from different sources
kept by the same database software.

Format: RIPE

Status: mandatory

changed:

Who and when changed this last.

Format: <email-address> YYMMDD

Example:
changed: johndoe@terabit-labs.nn 900401

Status: mandatory

remark:

Remarks/comments

Format: text string, multiple lines

Example:

remark: This is a group of institutes at WCW.

Status: optional

Appendix C — The idea of Autonomous System Macros.

AS Macros

For an AS that doesn’t use default routing it may be difficult to keep track of each and every new AS that is represented in the database. A convenient way around this is to define an ‘AS Macro’ which essentially is a convenient way to group or aggregate ASes. This would be done so that each and every AS guardian does not have to add a new AS to it’s as-in or as-out tags for it’s AS object.

However, it should be noted that this creates an implicit trust on the guardian of the AS-Macro.

The proposed idea would be to allow an AS-Macro as part of the <routing policy expression> for the "as-in" and "as-out" tags. The object could then be used as a list or group of ASes. So for example we could create an AS object as follows:

```
Example:

aut-num: AS786
as-in:   AS1755 100 AS-EBONE AND NOT AS1104
as-out   AS1755 AS786
```

Here is a summary of the proposed tags associated with the as-macro object itself:

```
as-macro:     [mandatory]
descr:        [mandatory]
as-list:      [mandatory]         [multiple]
guardian:     [mandatory]
tech-c:       [mandatory]         [multiple]
admin-c:      [mandatory]         [multiple]
remarks:      [optional]          [multiple]
changed:      [mandatory]         [multiple]
source:       [mandatory]         [RIPE]
```

Each item has the following syntax:

as-macro:

A macro containing at least two Autonomous Systems grouped together for ease of administration.

Format: AS-<String>, The string should be in upper case and not contain any special characters.

Example:

as-macro: AS-EBONE

Status: mandatory

as-list:

A list of ASes depicted by the macro.

Format: <aut-num> <aut-num> ...., mulitple. See Appendix A for <aut-num> definition.

Example:
as-list: AS786 AS513 AS1104

Status: mandatory

descr:

A short description of what the AS-macro represents.

Format: free text, one line per entry, multiple lines in sequence
Example:
descr: Ebone AS macro
descr: Details all ASes that make use of Ebone as a transit carrier

Stautus: mandatory

guardian:

Malibox of the guardian of the Autonomous system macro.

Format: <email-address>, single line.

Example:
guardian: as-ebone-guardian@ebone.net

Status: mandatory

tech-c:

Name of technical contact person.

Format: <firstname> <initials> <lastname>, multiple lines.
Example:
tech-c: John E. Doe

Status: mandatory

admin-c:

Name of administrative contact person.

Format: <firstname> <initials> <lastname>, multiple lines.
Example:
admin-c: Joe T. Bloggs

Status: mandatory

source: RIPE

Source of the information.
This is used to separate information from different sources

kept by the same database software.

Format: RIPE

Status: mandatory

changed:

Who and when changed this last.

Format: <email-address> YYMMDD

Example:

```
changed: johndoe@terabit-labs.nn 900401
```

Status: mandatory

remark:

Remarks/comments

Format: text string, multiple lines

Example:

```
remark: Pan-European backbone connecting several
remark: Regional Boundary Systems within Europe.
```

Status: optional

Appendix D — Transitioning from one AS to another.

As mentioned in Section 5, when changing service provider there will be a need for the network operator to move from one AS to another. Assuming that network operators use the RIPE database as proposed to maintain the needed routing filter lists, then there may be a synchronisation problem with the database image in use by the provider. The current frequency for database updates is once a day so if we look at this in terms of a transition, a clearly defined procedure is needed to make sure there is no loss of connectivity.

If we have the following scenario:

```
                           |
                        __|___          ______
+-------+             (      )        (      )
| NET X |-------------( AS A )----------( AS C )----
+-------+             (______)        (______)
                                          |
                                          |
                        ______          ___|__
                      (      )        (      )
                      ( AS B )----------( AS D )----
                      (______)        (______)
                           |
```

A network or even a group of networks, represented by NET X are connected to a single service provider, here shown as AS A. NET X has the same routing policy as AS A and as such has its network as-tag in the database as AS A. AS C, AS D and AS B for that matter all apply their own routing policies. Exactly what those polices are is not relevant for this discussion. However, when routing announcements from AS A regarding NET X are sent to AS C and beyond it is clear by using the database and the tools provided that NET X is part of AS A. The ASes will expect to see NET X as part of AS A's list of registered networks.

For whatever reason, the network operator of NET X decides to change its service provider from AS A to AS B. So the following transition takes place;

```
                           |
                        __|___          ______
+-------+             (      )        (      )
| NET X |- - - - - - -( AS A )----------( AS C )----
+-------+             (______)        (______)
     |                                    |
     |                                    |
     |                  ______          ___|__
     |                (      )        (      )
     +----------------( AS B )----------( AS D )----
                      (______)        (______)
                           |
```

NET X now connects to AS B. As soon as this is done this now means that NET X is announced as part of AS B as it now has the same routing policy as AS B. This is fine for AS B as it will presumably update the routing filter list because NET X is now a customer network. The problem is however, if AS D does not also update it routing filter list at the same time then it will not expect NET X announcements to originate from AS B. It will still think NET X will be part of AS A lists of networks and hence reject this routing information. This will be so until it has synchronised with the latest RIPE database. Even though the database update procedure is done on a scheduled basis this is obviously unacceptable. However, in reality this problem is not at bad as it may first seem. There is a simple solution to this and it is one that in practice would happen anyway. In the diagram above, the connection from NET X to AS A is shown as a dotted line. This is done intentionally to denote that most network operators would always factor in a small amount of transition time when changing service providers. With this assumption in mind (and the authors feel this is a fair one), then all that needs to be done whilst in transition is that AS A must continue to announce NET X to it's neighbors whilst AS B is also announcing NET X derived from the new connection. As the various ASes re-align with the RIPE database there would be a smooth transition in both routing and traffic from provider AS A to provider AS B. Obviously for this to take place, a small amount of coordination is needed between providers AS A and AS B but this is not an unreasonable expectation of any of todays service providers.

As the database is updated at least once every 24 hours then it should not take more than two days for most network operators to have aligned and all traffic to flow via the new connection. At this point AS A can stop its announcements of NET X and drop the connection as desired. So we finish up with this as a final configuration with of course no loss in connectivity for NET X;

```
                               |
                             __|___            ______
      +-------+             (      )          (      )
      | NET X |             ( AS A )----------( AS C )----
      +-------+             (______)          (______)
          |                                       |
          |                                       |
          |                  ______            ___|__
          |                 (      )          (      )
          +-----------------( AS B )----------( AS D )----
                            (______)          (______)
                               |
```

This simple procedure is what could be called a "Basic Provider Transition" and really should be followed by any conscientious service provider.

Appendix E — Routing Policy limitations

The generic example of a reasonable but un-implementable routing policy might be called "once joined means joined forever". Once traffic for the same destination network passes the same router, or the same AS at our level of abstraction, it will take exactly the same route to the destination (4).

In a concrete example AS Z might be connected to the outside world by two links. AS Z wishes to reserve these links for different kinds of traffic, let's call them black and white traffic. For this purpose the management of AS Z keeps two lists of ASes, the black and the white list. Together these lists comprise all ASes in the world reachable from AS Z.

```
     "W"
    <--->
...           AS Z .... NET 3
    <--->
     "B"
```

It is quite possible to implement the policy for traffic *originating* in AS Z: AS Z will only accept announcements for networks in white ASes on the white link and will only accept announcements for networks in black ASes on the black link. This causes traffic from networks within AS Z *towards* white ASes to use the white link and likewise traffic for black ASes to use the black link.

Note that this way of implementing things makes it necessary to decide on the colour of each new AS which appears before traffic can be sent to it from AS Z. A way around this would be to accept only white announcements via the white link and to accept all but white announcements on the black link. That way traffic from new ASes would automatically be sent down the black link and AS Z management would only need to keep the list of white ASes rather than two lists.

Now for the unimplementable part of the policy. This concerns traffic towards AS Z. Consider the following topology:

```
B AS ---)                    "W"
W AS ---)                    --->
B AS ---)>>  AS A  ---> ...           AS Z .... NET 3
B AS ---)                    --->
W AS ---)                    "B"
```

As seen from AS Z there are both black and white ASes "behind" AS A. Since ASes can make routing decisions based on destination only, AS A and all ASes between AS A and the two links connecting AS Z can only make the same decision for traffic directed at a network in AS Z, say NET 3. This means that traffic from both black and white ASes towards NET 3 will follow the same route once it passes through AS A. This will either be the black or the white route depending on the routing policies of AS A and all ASes between it and AS Z.

The important thing to note is that unless routing and forwarding decisions can be made based on both source and destination addresses, policies like the "black and white" example cannot be implemented in general because "once joined means joined forever".

(4) Disregarding special cases like "type of service" routing, load sharing and routing instabilities.