

Virtual Routing and Forwarding for Lightpaths Implementations at SARA

10 May 2011

Sander Boele, Pieter de Boer, Igor Idziejczak, Bas Kreukniet, Ronald van der Pol, Freek Dijkstra

SARA Computing and Networking Services

Abstract

In this paper we examine the options to implement strict IP routing policies, while maintaining flexibility to access services. A design based on static routes suffers from inflexibility. Multiple pieces of hardware are costly and take more administration than a single piece of hardware. With Virtual routing and forwarding (VRF) instances you can overcome these drawbacks: they provide strict use policies, flexibility in IP service assignment and scalability on a single router. We will describe two specific cases of VRF solutions as they are implemented at SARA.

Table of Contents

Introduction.....	3
Problem description	5
Solution.....	7
Implementations at SARA.....	11
SARA Grid VRFs	11
Vancis AMS-IX VRFs	15
Acknowledgements	20

Table of Figures

Figure 1. External connections to and from SARA.....	3
Figure 2. Example of a simple VRF router.....	8
Figure 3. GRID router with a sample of the configured VRFs	12
Figure 4. Full-routing without restrictions for the customer.....	15
Figure 5. VRFs for AMS-IX services on the Vancis routers.	17

Introduction

The demand for high capacity network connections is continuously increasing. Today, there is a strong demand for (semi-) private networking over high bandwidth connections. Often, Optical Private Networks (OPNs) are used. An OPN is a network that connects sites by way of permanent lightpaths. Companies often choose for an OPN for reasons of performance, capacity, cost, and occasionally privacy of their data. Lightpaths in an OPN typically carry dedicated data traffic, and should not carry regular Internet traffic.

In the scientific community, OPNs are often used to connect computing facilities, data storage environments and visualisation facilities with other computing centres and research institutes.

High bandwidth is needed because the equipment of some research institutes generates huge amounts of data that needs to be transported. Figure 1 depicts the 10 Gb/s network connections to the Dutch e-Science centre SARA as of late 2010. SARA provides the same services to different users.

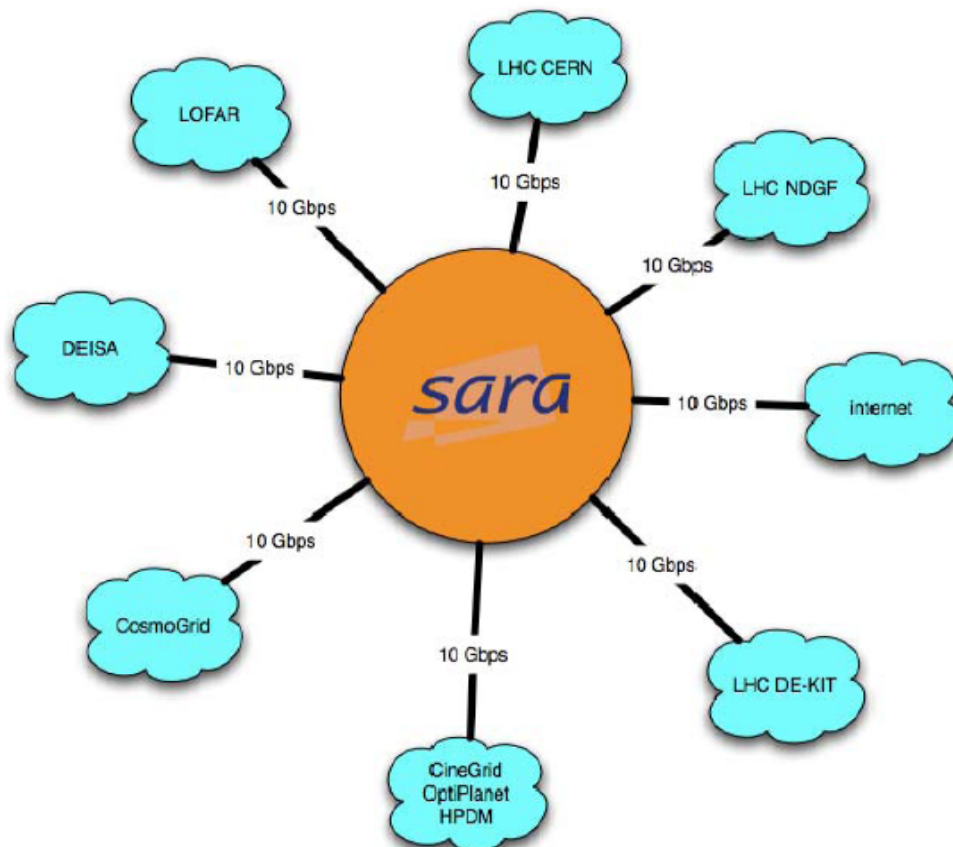


Figure 1. External connections to and from SARA.

The astrophysics LOFAR project and the nuclear physics LHC project connect remote instruments to storage facilities at SARA. The DEISA, and CosmoGrid projects connect supercomputing centres. The CineGrid, OptiPlanet and HPDM projects connect visualisation facilities.

For example, the storage facility is connected to the LOFAR connection, three LHC connections and the public internet. A design decision was to create a single storage facility to offer scalable services to all projects instead of creating individual storage facilities for each project. A consequence is that

all storage OPNs should access the same facility. An OPN is a layer 1 or layer 2 network. For reasons of scalability and security it is not possible to connect all OPNs to the same VLAN. It is a requirement of SARA to make a clear distinction between administrative domains. Moreover, some OPNs have policies that prohibit routing from and to other OPNs. To accommodate these policies, SARA decided on an architecture where all lightpaths are terminated on a router, and all services (storage, computing, and visualisation) are reachable from this router. The separation of the traffic is made at layer 3 as we will see in the next paragraphs.

Problem description

Operators of Optical Private Networks (OPNs) demand a strict routing policy: only traffic sourced from or destined to a particular service (be it a cluster, supercomputer, storage facility or single user) is allowed to make use of that OPN. All other traffic must be routed through other paths (for example the public internet). This poses the problem of segregating the privileged traffic allowed to traverse the OPN from the unprivileged traffic that is not allowed to traverse the OPN.

Until five years ago many organisations did not use OPNs but simply used the public internet to provide connectivity between their services. This solution did not address many threats and problems: a strict routing policy was by definition not possible because destination based routing is used.

A more secure option for privileged traffic is to terminate each OPN on a dedicated router, and separate IP routing on a per-router basis. Each dedicated router is used to provide a network connection to one service at the service provider (e.g. SARA). Specific routing policies are maintained on a per-router basis for every service. In addition, the router may have a default gateway to the public internet or a campus network.

This solution has the drawback that it does not scale. First you need a lot of routers. Although the layer 3 requirements for an OPN are typically very light (usually, only a few IP blocks need to be known in an OPN). Second, you will need to make a lot of physical connections from all these routers to the desired services. Therefore, buying many routers seems like a waste of money. Apart from that, it will also take a lot of administrative hustle.

Segregating traffic from different OPNs is a challenge in itself. With traditional destination based routing there is no additional policy configured. The routing makes use of two principles: best path principles and the matching of the most specific destination prefix. Each routing protocol has a set of rules to determine the 'best' path to the destination. This is never based on the source prefix of the traffic. Looking at OPN technology, we don't want the best routes to be learned to the destination, we want to keep all data from the connected services within that OPN. So, source addresses should be evaluated. If they are allowed they will be made available in the routing tables within a specific OPN. If they are disallowed, these addresses should simply not be known in the OPN.

A well-known routing problem (particularly in the public internet) is the problem of more specific routes: when a router uses a dynamic routing protocol it often learns many routes to the same destination. A basic principle of routing is that more specific routes are preferred over less specific ones. Even if a path is longer or the preferences are higher, the route with the most specific prefix will be elected over a less specific one. Sometimes this must be avoided, for instance when the more specific prefix comes from an expensive upstream link. You want to minimize the traffic over this link to save money. Another problem with more specifics can be as follows: An OPN does not have the most specific routes to its remote service. Imagine a router learns a more specific route via the public internet than the one it has learned from its remote side via the OPN. The router will choose the public internet to transfer the data between the services. This is definitely what must be avoided!

A solution can be provided with Virtual Routing and Forwarding (VRF). In the next paragraphs we will explain this technology, reveal its advantages and show some specific examples at SARA¹.

¹ Examples are provided in order to explain how routing between VRFs is structured and manipulated. The examples differ from the actual configurations at the moment of writing.

Solution

VRF is a technology that allows a single physical router to be divided into multiple virtual routers sharing the physical resources. One of the main benefits of VRFs is the ability to create multiple routing policies not only based on the destination of the traffic, but also on the source of the traffic. The key to the VRF technology is that each VRF instance has its own routing table. Between VRFs import and export policies allow routes to be transferred from one routing table to another. With these policies, routes can also be modified during this transfer, e.g. prepending a community or adjusting a metric. When only a small number of routes is exported, we speak of 'route-leaking'. That means all routes are blocked except a few, which are explicitly selected to provide routing between specific sources and destinations. Using VRFs allows you to have multiple routing tables at the same time on one physical router. The routing instances are independent. Each protocol you want to run in a VRF needs to be configured locally. Enabling a protocol in a VRF will create a new process for that protocol and only for that VRF. So, for BGP, there will be neighbor and BGP databases per VRF. For OSPF, there will be topology and tree databases per VRF. In addition, a VRF has its own forwarding table that designates the next-hop for each data packet. These tables prevent traffic from being forwarded outside a specific VRF path and also keep out traffic that should remain outside that VRF. Due to this strict separation of routing, you could even use overlapping IP addresses with other VRFs. Virtual routing requires minimal changes to the existing network infrastructure and is a cheap and easy solution. Moreover, administration is simple, compared to using multiple physical routers.

Figure 2 illustrates the basic principles of VRF technology by means of a simple VRF setup. Imagine two OPNs that need access to a storage cluster, OPN-A needs full access to the storage cluster (in this example 145.100.32.0/22). OPN-B only needs access to a small subset of the storage systems (in this example 145.100.32.0/29). Second, the storage cluster needs access to both OPNs and to the public internet.

The configuration for OPN-A will look like this²:

```
routing-instances {
  OPN-A {
    instance-type vrf;
    interface ge-0/1/1
      route-distinguisher 1126:111;
  }
  vrf-export export-OPN-A-routes-to-storage-cluster
  routing-options {
    static route 145.100.32.0/22 next-table storage-cluster
  }
}
```

By issuing the command '**set routing-instances OPN-A instance-type vrf**', a VRF is created named 'OPN-A'. We need to bring the interface of the OPN-A in the VRF. This is configured with the line: **interface ge-0/1/1**.

² The configuration examples are in JUNOS (the Juniper Networks operating system).

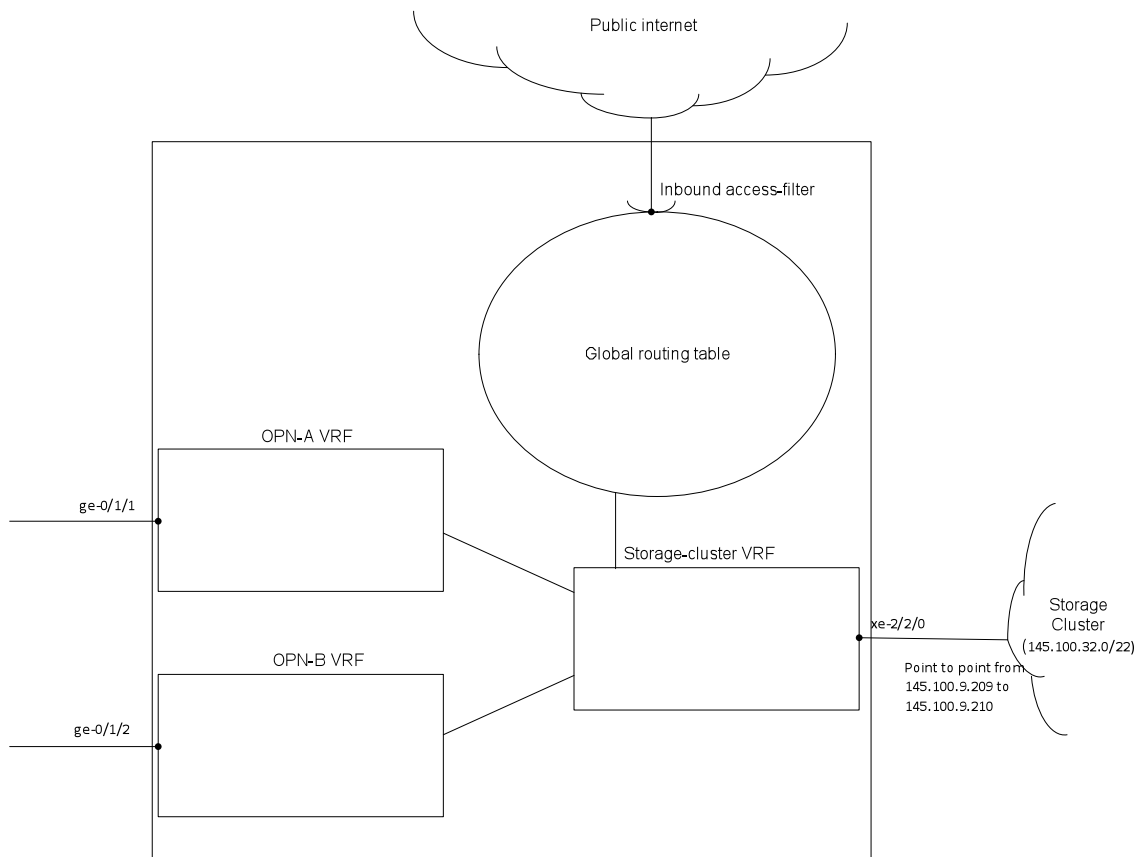


Figure 2. Example of a simple VRF router

We export all the routes of this OPN to the “storage-cluster” VRF with an export policy, named “export-OPN-A-routes-to-storage-cluster” (specification of policies is not included here, for examples see the paragraph “SARA Grid VRFs”) and we import them with an import policy in the “storage-cluster VRF” (see next paragraphs).

To learn routes from the storage-cluster VRF in other VRFs, we decide for static route leaking instead of an export and import policy. Static route leaking is configured in the receiving OPN. So in this case, we configure this in OPN-A and OP-B. For OPN-A, a static route with the **next-table** option is written in OPN-A. Do not confuse this with a normal static route (normal static routes make use of the **next-hop** option).

The purpose of this static is that we want to give the OPN-A selective access to the storage-cluster VRF”. By this static, any queries for this route in the OPN-A VRF are directed to the storage-cluster VRF. The OPN will not find any other routes that are in the storage-cluster VRF. Now, imagine, we would like to give OPN-B selective access to only a small block of addresses within the storage cluster.

```

routing-instances {
  OPN-B {
    instance-type vrf;
    interface ge-0/1/2;
    route-distinguisher 1126:222;
    vrf-export export-OPN-B-routes-to-storagecluster;
    routing-options {
      static route 145.100.32.0/29 next-table storage-cluster ;
    }
  }
}

```


This is easily achieved by configuring another instance of static route leaking: in the VRF for OPN-B we write: `static route 145.100.32.0/29 next-table storage-cluster`. Note that OPN-A and OPN-B have overlapping routing information. This does not create any problem with well-configured VRFs.

The VRF for the storage cluster will look like this:

```
edit routing-instances {
  storage-cluster {
    instance-type vrf;
    interface xe-2/2/0
    route-distinguisher 1126:999;

    vrf-import import-opn-a-and-opn-b

    routing-options {
      static route 145.100.32.0/22 next-hop 145.100.9.210
      static route 0.0.0.0/0 next-table inet.0
    }
  }
}
```

Note, earlier, we set an export policy in OPN A and OPN B VRF. The export policies do not inject routes directly into an other VRF, it only makes them available to be imported by an import policy. The export policy adds attributes to the exported routes, while the import policy takes a copy of them when match criteria are met. We enable a policy to import all routes from OPN-A VRF and OPN-B VRF (called 'import-opn-a-and-opn-b') in the storage-cluster VRF:

```
policy-statement export-OPN-A-routes-to-storage-cluster {
  then {
    community add opn;
    accept;
  }
}
policy-statement export-OPN-B-routes-to-storage-cluster {
  then {
    community add opn;
    accept;
  }
}
policy-statement import-opn-a-and-opn-b {
  from community opn;
  then accept;
}
```

We set a route to the storage cluster: `static route 145.100.32.0/22 next-hop 145.100.9.210`. For the sake of simplicity we use a static route in the example, but any IP routing protocol can be used here.

Finally we set a static default route to the global routing table (inet.0): `static route 0.0.0.0/0 next-table inet.0`. We prefer a default route to inet.0 rather than an import policy from the inet.0 since the latter will cause the huge amount of routes from the global table to be imported in the storage-cluster VRF. This route is not exported to the OPN VRFs since we only leak storage-cluster IP blocks. By setting the default to the global routing table, the storage-cluster VRF has routes to all sites.

The global configuration, on its turn, must have a static route for the storage cluster to provide routing from the public internet to the storage systems. This static is the same as the one we saw in OPN-A:

```
static route 145.100.32.0/22 next-table storage-cluster
```

This static route is configured globally since the global routing table is not a VRF.

The uplink to the public internet is protected with an access-list. This restricts unlimited access to the storage systems from the outside. Note that this is not a special requirement of the VRF set-up: you would have needed this access-list anyway. For the VRFs no additional security is necessary to protect it from the outside.

IP information is imported and exported and these routes can be modified as well. To ensure limited access to a few sources, we only leak routes. No changes are needed elsewhere in the local network. Packets are routed between the VRFs on the router, while the VRFs forward packets themselves. Hosts in the local (storage cluster) network can access the global routing table for external connectivity and can acquire selective access to specific OPNs. In the next paragraphs, we go on to explain two more VRF examples. These are VRF solutions at SARA and VANCIS. They reveal elaborate ways of route leaking. The first example uses static route leaking, while the second uses dynamic route leaking.

Implementations at SARA

SARA Grid VRFs

SARA makes a clear distinction between their storage and computing services.

The SARA grid storage cluster is used for data storage by several external grid compute clusters in the national and international scientific community. The challenge is that the different connected external compute clusters share a general IP-destination, the storage cluster, while at the same time compute clusters should not see each other. It is necessary that the storage cluster is accessible from the outside, either via the public internet, or via OPNs. Since the total number of customers is low, we decided to use only one router for the connections to SARA's grid storage services. We chose a Juniper MX960. The choice for a single router challenged us to do virtual routing in an elegant and secure way.

The SARA grid storage cluster is used in the Large Hadron Collider (LHC) project at CERN in Switzerland and France. For data transmission between CERN and the SARA grid storage cluster, the Large Hadron Collider Optical Private Network (LHCOPN) is used. The LHCOPN is a dedicated private network of multiple lightpaths connecting international networks³ with each other. The lightpaths of the LHCOPN are connected to the grid router in a LHCOPN VRF. E-BGP sessions with LHCOPN peers are used to advertise and receive prefixes. This implies that the LHCOPN VRF learns the routes of the LHCOPN network. The next step is that they are exported to a VRF called "storagecluster-shared". This VRF should have routes to all sites including the public internet.

There is a local VRF for network monitoring. This one is called "LHCOPN-monitoring" in Figure 3. This network should connect from and to the LHCOPN and should also have access to the public internet.

The SARA grid storage cluster is used by other compute clusters in the Netherlands as well, for example LOFAR and Life Science Grid (LSG). Traffic to and from these clusters also uses dedicated light paths. They are not allowed to have access to the LHCOPN but they need access to the grid storage cluster. For an even more enhanced security than a standard VRF solution provides (like Figure 2), we decided to configure different paths between VRFs for each direction of traffic. Traffic flows are displayed in Figure 3.

³ For a complete view of the LHCOPN, see: <https://twiki.cern.ch/twiki/bin/view/LHCOPN/OverallNetworkMaps>


```

edit policy-options {
    policy-statement lofar-export {
        then {
            community add lofar;
            accept;
        }
    }
}

```

and we formulate a community with a unique value:

```
community lofar members target:1126:3;
```

Next, we configure the export policy in the corresponding VRF:

```

edit routing-instances {
    LOFAR {
        instance-type vrf;
        interface xe-1/3/0.0;
        route-distinguisher 1126:3;
        vrf-export lofar-export;
    }
}

```

This will export all the routes from the LOFAR routing table.

An import policy is required to match the above export policy. The import policy that gets the routes in another VRF is configured like this:

```

policy-statement lofar-import {
    from community lofar;
    then accept;
}

```

And we place this policy VRF-import in the storage cluster-shared VRF:

```

routing-instances {
    storagecluster-shared {
        instance-type vrf;
        route-distinguisher 1126:999;
        vrf-import [ storagecluster-import lofar-import ];
    }
}

```

The storagecluster-shared VRF is also configured to import LHCOPN routes (not included in example). This VRF learns all routes from the LSG, LOFAR and LHCOPN VRFs. The only criterion for importing routes is the community that is prepended to the routes coming from a VRF. However, from the side of these externally connected VRFs static route leaking is used to find the way to the storage cluster. These VRFs have static routes that selectively tell that the route to the storage cluster is found via the global routing table. The configuration is as follows:

```

routing-instances {
    LOFAR {
        instance-type vrf;

        routing-options {

```

```

        rib LOFAR.inet.0 {
            static {
                route 145.100.32.0/22 next-table inet.0;
            }
        }
    }
}

```

Instead of a next-hop option, we use a next-table option in order to direct to another routing table. In this case this is the global (internet) routing table. In the global routing table, a second route lookup is performed in order to find the interface to route the packets out.).

A route to the storage cluster in the global routing table is injected by means of a simple static at the global level of the configuration:

```

edit routing-options {
    static {
        route 145.100.32.0/22 next-hop 145.100.9.210;
    }
}

```

In order to provide routing from the storage cluster to the VRFs we configured a way to access the “storagecluster-shared” directly for route lookups. The interface from the storage cluster is connected to the VRF router at the global level. The interface has an inbound filter, configured as follows:

```

filter storage-lan-in {
    term use-storagecluster-shared {
        then {
            routing-instance storagecluster-shared;
        }
    }
}

```

Since no ‘from’ statement is set, this means that *any* traffic will be redirected to the storagecluster-shared VRF. When traffic arrives at the interface, it will be passed to the storagecluster-shared VRF immediately. Nothing will be done at the global level of the router. In the storagecluster-shared VRF a route lookup is performed and traffic will be directed to the correct interface out in the corresponding VRF. Because the storage cluster should be able to access the public internet, there is also a default route in the storagecluster-shared VRF that points back to the global routing table. Again, a static route with the next-table option is used:

```

routing-instances {
    storagecluster-shared {
        instance-type vrf;
    }
    (..)
    routing-options {
        static {
            route 0.0.0.0/0 next-table inet.0;
        }
    }
}

```

Vancis AMS-IX VRFs

Vancis, a subsidiary company of SARA, sells internet connectivity to customers. To do so, Vancis has links to upstream providers and VANCIS maintains a connection to the Amsterdam Internet Exchange (AMS-IX). A special service that Vancis offers to some of its customers is the AMS-IX only variant. In this variant a customer buys only connectivity to AMS-IX peers connected to VANCIS.

The upstream providers typically provide full routing. All internet routes known to them are advertised to their peers. The AMS-IX allows peers to exchange their own routes only. For that reason an export filter must be set on the peer (or peer group) with the AMS-IX. All routes learned from the upstream or from anywhere else outside the network of the ISP and its (directly) connected customers, must be filtered out. At the links to upstream providers, we encounter a similar situation: the upstream providers do not want to receive the routes an ISP learns over the AMS-IX. Here, an export filter is set at the peer level as well. Usually, both measures are realized by prepending a community on the receipt of a route. The tagged routes are rejected for advertisement by filtering on this community-tag. Figure 4 denotes these filters with a '(' symbol.

The example in Figure 4 shows a setup where a customer is serviced with unrestricted internet access including the routes learned from the upstream providers and the routes learned over the AMS-IX peers. As long as there are no filters towards the customer they will receive all routes. Similarly, VANCIS advertises the routes received from the customer over the AMS-IX peers and over the upstream peers. This is not yet the desired situation for a customer paying for AMS-IX only service.

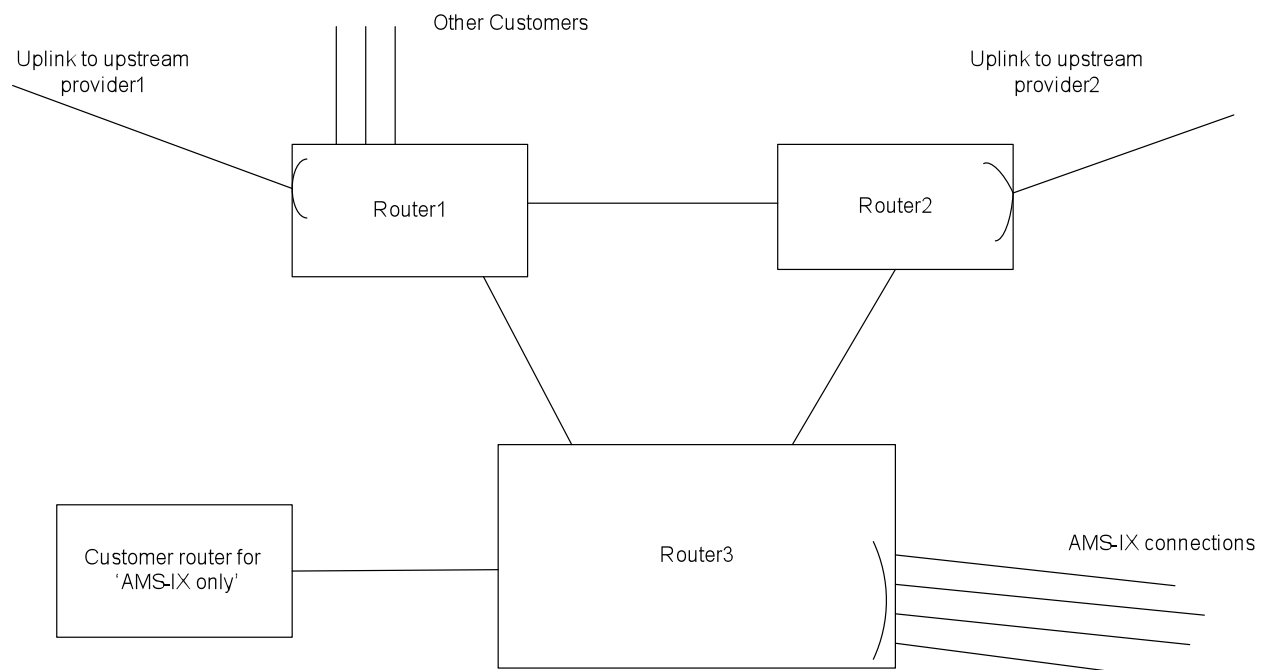


Figure 4. Full-routing without restrictions for the customer.

The challenge in this example is that routes received over the AMS-IX are not necessarily the best and most specific while the customer must use the AMS-IX anyway. In our ISP environment we have seen that approximately 10,000 prefixes received via our direct connection with our upstream providers are more specific than those received over the AMS-IX peerings. Since traffic over the

uplinks is much more costly than over the AMS-IX peerings, some customers choose for a cheaper solution: only use the AMS-IX peers. Could this be established without VRFs while we keep this architecture? ?

Let's consider tagging all routes learned over the upstream peers with a community. At Router 3 we place an export filter towards the customer router in order to filter these routes.

Routes learned over the AMS-IX peers are not tagged with a community. Routes learned via the upstream are tagged with a community say 1126:101. Now, imagine, we receive route 192.0.2.0/24 from one of the AMS-IX peers and route 192.0.2.128/25 from the upstream provider. In the routing table of the VANCIS router (Router 3), these routes coexist in the same table. This must be this way since VANCIS and some other customers always want the best path⁴; therefore the routes must be made available at the VANCIS systems. Before advertising routes to the AMS-IX transit customer, we filter all routes with the community 1126:101. The 192.0.2.0/24 learned over the AMS-IX is the only one sent to the customer router, and this route is inserted in their routing table.

Imagine, the customer wants to reach 192.0.2.129. His routing table will only have the 192.0.2.0/24. The route will be a BGP learned route and point to its EBGP peer. That is Router 3 in Figure 5. When the IP packet arrives here, a new route lookup will be performed and this time, the 'better' route, the 192.0.2.128/25, is found. Since we do not filter anymore on communities from here on, the customer packet will also take the path via the upstream despite this destination prefix was not known to him, initially. This will no work with BGP filters only and without VRFs.

We decided to segregate routing by means of VRFs. Since the routes in the public internet are constantly changing, the new setup must be able to continuously adapt to new changes. We found that the creation of two VRFs is sufficient to address the problem. On Router 3, we created a VRF called "AMS-IX" and a VRF called "all-routes", see figure 6.

⁴ The drawing does not list all connections to the VANCIS router. In reality there are more links to the routers, than shown here.

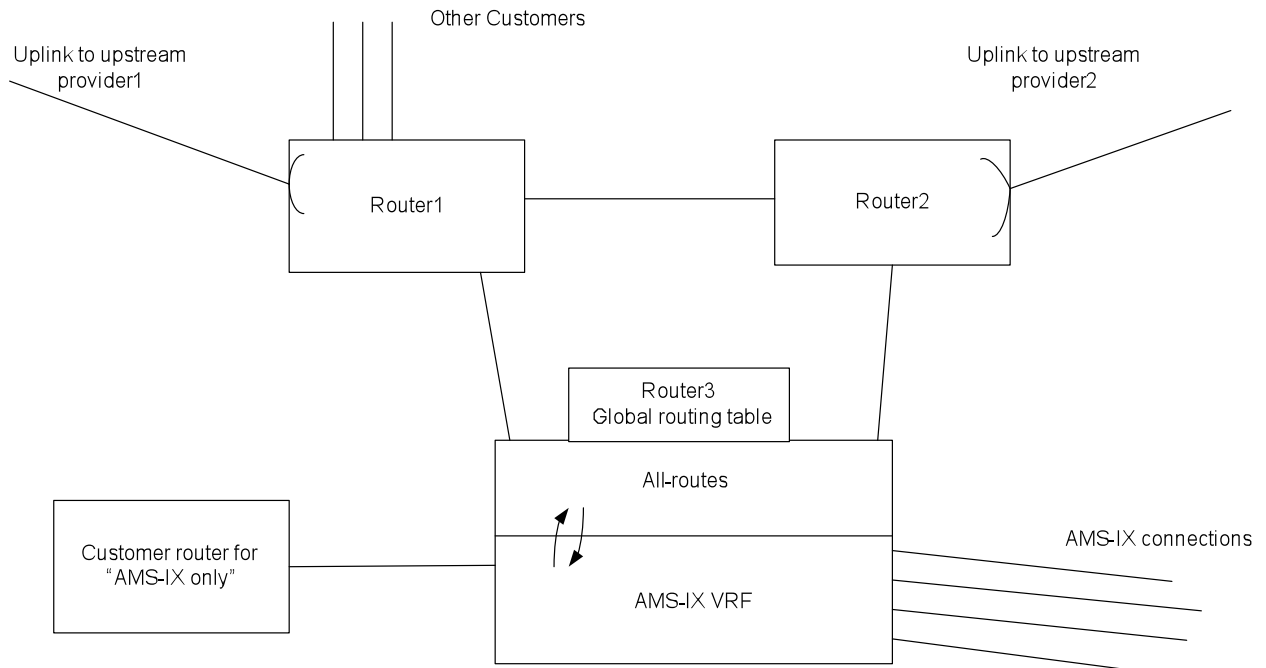


Figure 5. VRFs for AMS-IX services on the Vancis routers. VRFs are created on Router 3, called “AMS-IX” and “all-routes”. Arrows indicate exchange of routes by way of export and import policies between the routing instances⁵.

The connection from the customer with AMS-IX-only is moved into the AMS-IX VRF including the BGP peering. All AMS-IX connections including the peering configurations are moved into the AMS-IX VRF.

All the routes we learn from AMS-IX peers are tagged with a community 1126:102. All routes from the customer with the AMS-IX-only service are tagged with a community 1126:103. An export and an import policy is created in order to bring those routes from the AMS-IX VRF into the all-routes VRF. All routes from the AMS-IX VRF are exported into the global routing table; there are no restrictions from the AMS-IX VRF to the all-routes VRF. This ensures that the routes are present and can be advertised within the rest of the VANCIS network and VANCIS Customer networks. The prefixes from the Customer “AMS-IX only” are filtered out with an export policy at the peers to the upstream providers at Router 1 and Router 2. This export policy is configured as follows:

```

policy-statement export-all-except-ams-ix-and-ams-ix-only-customer {
  term no-advertise-routes-from-upstream {
    from {
      community [ams-ix ams-ix-only-customer]
    }
    then {
      reject;
    }

    term anything-else {
      then {
        accept;
      }
    }
  }
}

```

⁵ Note that the global routing table has no connected interfaces. Since all routing is done with the VRFs, this table is empty and does not have any function in routing anymore.

```

    }
}

policy-options { community ams-ix members 1126:102 }
policy-options { community ams-ix-only-customer members 1126:103 }

```

The AMS-IX community (1126:102) was already configured to be rejected. With the creation of the AMS-IX VRF, only the customer community (1126:103) was added.

Now, let's examine which routes we like to have in the AMS-IX VRF. The routes from the upstream should not be known here. How do we import routes from the all-routes VRF into the AMS-IX VRF, except the ones learned from the upstream peers? The routes from the upstream peers are already tagged with a community (1126:101). First we set an export policy in the all routes VRF that exports all routes. This is a simple policy that looks like this:

```

policy-statement export-all-routes {
  term export-all {
    then {
      accept;
    }
  }
}

```

Since no "from" statement is set, all routes will be elected for export.

The trick is done at the import filter in the AMS-IX VRF. The upstream routes are easily filtered out with this policy:

```

policy-statement import-all-except-upstreamroutes {
  term no-advertise-routes-from-upstream {
    from {
      community upstream
    }
    then {
      reject;
    }
  }
  term anything-else {
    then {
      accept;
    }
  }
}

policy-options { community upstream members 1126:101 }

```

When this import policy is set in the AMS-IX VRF, the original filter at the peer level to the AMS-IX (see Figure 4), is no more necessary and can be deleted.

With those steps we build the bridge between the all-routes VRF and the AMS-IX VRF in such a way that the table in the all-routes VRF holds all the routes from the AMS-IX-only customer. The AMS-IX VRF, instead, does not contain the EBGP learned routes from the upstream. Thus we have eliminated the problem of more specific prefixes (the route 192.0.2.128/25 in the example above will not be available in the routing table of the AMS-IX VRF).

Note that the route filtering works in *real-time*. Since the communities are prepended at the receipt of BGP learned routes, the policies take any newly learned route into account, while arriving over the same link. Thus, this provides a method for dynamic route leaking.

Acknowledgements

This work was funded in part by the GÉANT3 project.

The authors like to thank Peter Tavenier and Ingrid Janssen for proofreading this document, and Hanno Pet for contributing to an early draft of this document.