

A new VPN routing approach for large scale networks

Zied Ben Houidi

Orange Labs and UPMC Sorbonne Universites
zied.benhoudi@orange-ftgroup.com

Mickael Meulle

Orange Labs
michael.meulle@orange-ftgroup.com

Abstract—One of the most common provider provisioned VPN technologies uses MPLS as a data plane for customer flow isolation and BGP as a control plane for routing between VPN sites. From a data plane perspective, such networks can provision hundreds of thousands of VPN sites. However, the BGP control plane is prone to scalability concerns. Some BGP routers in VPN backbones must handle routes for all the VPN sites that the provider connects. The number of sites can generate two million BGP routes in large VPN backbones, almost ten times the number of routes in a core Internet router. Prior work proposed solutions to evolve such networks. Yet, we argue that they fail to address the root cause of VPN routing performance issues. In this paper, we show that VPN routing scheme’s poor scalability stems from the application to VPNs of a protocol originally designed for full routing, specifically the Internet. Rather than evolving the current standard based on BGP, we take a principled approach to rethink routing in large VPNs. We propose Two-Step VPN Routing, a new approach for scalable VPN routing. We validate our design choices and compare our approach to existing ones, using both BGP updates and router configurations collected from a large VPN provider.

VPN sites is threatened by slow BGP convergence, extremely large BGP routing tables and slow BGP table transfers [6], [7], [8]. Provider routers use BGP to exchange VPN customer routes with each other. Since a full mesh of BGP sessions between BGP routers becomes infeasible as their number increases, VPN operators often use route reflectors to scale the BGP route distribution. However, with the ever-increasing number of VPN sites, this technique cannot contain routing table size explosion. In fact, the number of BGP routes in such networks may reach around 2 million routes [7], which is one order of magnitude higher than the number of BGP routes in an Internet core router. This high figure arises fundamentally from the impossibility to aggregate customer routes.

In order to cope with these scalability issues, prior work on BGP MPLS VPNs proposes incremental solutions: RT-constraints [9] makes route reflectors maintain the state only for routes they need for their client routers (instead of all routes). Multiple plane RR [7] partitions the BGP control plane, each part handling only a subset of the routes. Although they can be helpful, we argue that they are not sufficient. Either they do not reduce enough the routing table size (RT-constraints) or add extra costs and configuration complexity (multiple plane RR). Existing proposals aim at circumventing the problem but fail to address the root cause of it, namely the original design of BGP and its inadequacy with regard to VPN routing specific requirements. Indeed, BGP was designed for Internet backbones where BGP speakers need to perform full routing. Transposed to VPNs, BGP therefore causes some routers to keep the state for all routes of all VPN sites, eventually leading to major scalability concerns. This is why we consider BGP not appropriate for VPN routing in large networks.

Drawing the proper conclusion from this claim, we choose in this paper to rethink the VPN routing approach for large networks, rather than evolving the current standard. We start by providing in Sec. II a simple model, that captures provider provisioned VPNs basic concepts. We first use this model to specify the basic routing requirements of such networks and then show in Sec. III how BGP is coerced in the BGP MPLS IP VPN standard to serve those requirements. We then give an overview of prior efforts in BGP optimization for VPN routing and discuss their limitations in Sec. IV-A. We demonstrate in Sec. IV-B that BGP “Hop-by-Hop signaling” is responsible for poor VPN routing scalability and hence that BGP is not the

I. INTRODUCTION

Enterprises today often have sites spread across distant locations that need to interconnect. Instead of having fully dedicated links between their sites, many enterprises prefer to contract a Virtual Private Network (VPN) service from a VPN service provider, reducing thereby the connection costs. This service model is known as the *provider provisioned VPN service*. In this model, the VPN provider shares its physical network infrastructure among multiple enterprises, guaranteeing isolation of virtual networks.

There has been a tremendous standardization effort within the IETF to specify routing mechanisms for provider provisioned VPNs [1], [2], [3], [4], [5]. Among the proposed standards, BGP MPLS IP VPNs [3] stand as the most popular. This technology uses BGP as a control plane to provide VPN routing and MPLS as a transport technique to achieve isolation between customer traffic. Its popularity stems from the high number of customers supported (thousands of customers and hundreds of thousands of VPN sites), but also from the fact that IP layer 3 VPNs are easier to manage for customers.

From a provider perspective, BGP MPLS IP VPNs allow for quick and easy provisioning of new customers. This operational ease, however, is compromised by routing scalability and convergence issues for major providers. Indeed, performance of provider networks that host a large number of

best fit for VPN routing. We therefore propose in Sec. V Two-Step VPN Routing, a scalable End-to-End signaling approach to serve the routing requirements derived from our model. This alternative approach is based on separating two concerns, VPN topology creation and maintenance from VPN route advertisement, in order to remove useless routing state from core routers. Finally, in Sec. VI, we justify our design choices based on two months of BGP VPN routing data and router configurations collected from a large VPN provider housing hundreds of border routers connected to hundreds of thousands of VPN sites. Coupling the router configurations and the VPN routing data allows us to predict the behavior of Two-Step VPN Routing and compare it to state-of-the-art approaches to scale VPN routing.

II. VPN ROUTING: PROBLEM DEFINITION

In this section, we present a simple model for VPN service provisioning and use it to derive the basic VPN routing needs. We first define the main goals of provider provisioned VPNs.

A. Provider goals and desired properties

Isolation is a key word in the VPN service provisioning. The different VPNs have to be isolated as if they had dedicated links between their different sites. We define two levels of isolation, control plane isolation and data plane isolation.

- VPN control plane isolation concerns both signaling and addressing. Both should be isolated. Each VPN can have its own addressing and different VPNs can have overlapping addresses. Also, signaling of one VPN must not affect signaling of another VPN.
- Data plane isolation concerns performances and privacy. The performances of one VPN should not be affected by the fact that other VPNs share the same infrastructure.

B. A simple provider network model

We model the shared network, offered by the provider, by a set of *interfaces* through which VPN *sites* are connected. Each interface connects one VPN site. A VPN is composed of a set of sites that need to be interconnected. A VPN site is identified by one (or more) *site identifier(s)*, S_i . A site identifier is local to the VPN, that is, different sites belonging to different VPNs can have the same identifiers. For clarity, we consider in this section the case of one S_i per site but the analysis can be extended to more than one S_i .

Each interface I_i has an interface locator $I_{loc\ i}$. We assume that from any source interface, I_s , it is possible for the provider network to provide a *tunnel* and reach any destination interface I_d just by knowing its interface locator $I_{loc\ d}$. A *tunnel* is a logical link between the two interfaces. We assume that the tunnel provides the data plane isolation defined in Sec. II-A. We also assume that each interface has a *VPN interface manager (IM)* that handles all the operations concerning that interface.

This provider model can be extended to take into consideration classes of service (CoS) by assuming that an interface I_d can have more than one unique interface locator each

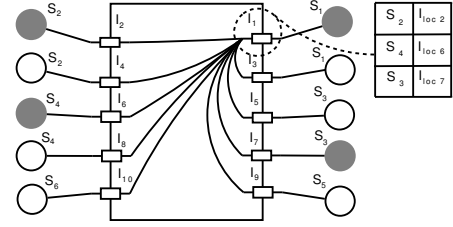


Fig. 1. Example of a provider network connecting two VPNs (resp. in white and gray). For the site S_1 to send a message to S_2 , the VPN interface manager (IM) of I_1 only needs to know the destination interface locator corresponding to S_2 . The message shall go through the virtual link (I_1, I_2) .

corresponding to one particular CoS. Fig. 1 provides a very simple example of a provider provisioned VPN model, to illustrate the terms.

C. Basic VPN routing needs

We now specify the basic *routing* needs for provider provisioned VPNs under this model. By *Routing*, we mean the set of procedures that allow a source host to reach a destination host. By definition, distant VPN sites that want to communicate do not have direct connectivity. Therefore, routing is done by the VPN provider. Similar to the IM of I_1 in Fig. 1, each IM in the provider network needs a *mapping between interface locators and identifiers of sites that belong to the same VPN*. Building this mapping constitutes the basic routing need for provider provisioned VPNs.

Given these mappings, there are two approaches for a VPN interface manager to forward VPN customer packets. The first is to have a routing and a forwarding table per VPN interface and forward packets based on their destination address. The second is to use encapsulation as with map-and-encap schemes [10], [11]. In this paper, we focus on the first set of approaches and leave map-and-encap approaches for VPN routing for future work. Following the first set of approaches, the mapping (site id, interface locator) is translated to a routing table entry.

There are basically two ways to build a routing table, link state or distance/path vector. A link state approach is not suitable for VPN routing because it can not scale to hundreds of thousands of (VPN site, interface) links that a VPN network needs to provision. Using a distance vector approach, each VPN interface manager needs to send its mappings (routes) to all distant VPN interface managers that belong to the same VPN. This procedure contains the following *two tasks*:

- 1) Knowing which interfaces belong to the same VPN.
- 2) Exchanging mapping information only between these interfaces in order to build the mapping.

It is important to note that our model is compliant with the integrity constraints defined by Bush et al. [12], since it assumes that a VPN interface connects a unique VPN site (or a set of sites that belong to the same VPN).

III. VPN ROUTING WITH BGP MPLS IP VPNs

This section shows how BGP MPLS IP VPNs use BGP to implement the basic routing needs described in Sec. II.

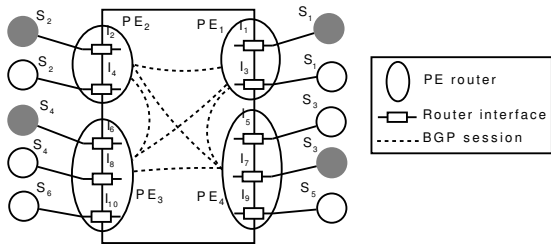


Fig. 2. Equivalence with the model

A. Equivalence with our model

With BGP MPLS VPNs, a provider backbone has a set of provider edge (PE) routers that connect VPN sites. The data plane between PEs is based on IP and MPLS. From any PE in the backbone, it is possible to provide an MPLS tunnel to any other PE. Fig. 2 presents the equivalent using our model of a BGP MPLS VPN provider backbone that has four PE routers to connect 10 VPN sites and two VPN customers. With respect to our model, site ids are IP addresses of the VPN sites. Each PE router has a set of physical interfaces that connect each a VPN customer site. The interface locator locates an interface inside the provider backbone. With BGP MPLS VPNs, locating an interface is done using two identifiers, one that locates the PE inside the provider backbone and one that locates the interface inside the PE. This is done in some implementations using two MPLS labels, one external that identifies the PE in the provider backbone and one internal that identifies the interface inside the PE.

The VPN interface manager task is done in a "centralized" way by the PE that hosts the interface. In fact, each PE associates a virtual routing and forwarding (VRF) table per interface¹. The VRF table contains the mapping information necessary for routing. In BGP MPLS VPNs words, a VRF table entry of a PE router PE_{source} contains the following information: IP_{dest} , $MPLS\ label\ of\ PE_{dest}$, $MPLS\ internal\ label$. The first label is used to tunnel the data till PE_{dest} and the second to reach the destination interface and therefore the VPN site. The aim of a VPN routing protocol is to build this mapping and populate the VRF table.

B. Encoding the mapping information in BGP routes

We identified in Sec. II-C two basic tasks to build the mapping, (1) knowing which interfaces belong to the same VPN and (2) sending mapping information only to those interfaces. With BGP MPLS VPNs, the two tasks are performed at the same time using multi protocol extensions of the border gateway protocol (BGP) [13]. BGP is responsible for encoding and flooding the mapping among the PEs of the provider backbone. The VPN provider backbone runs BGP between all its PE routers. This is represented by the BGP sessions in Fig. 2. The four PEs have a full mesh of BGP sessions in order to exchange BGP routes containing the mapping information necessary for VPN routing.

¹In practice, a VRF can be associated to a set of interfaces that connect customer routers belonging to the same VPN site

The mapping (Site IP, PE address or label, internal label) corresponding to a VPN interface is encoded in a BGP route. The PE address is stored in the Next Hop attribute. The site IP together with the internal label are encoded in the BGP network layer reachability information (NLRI) field. However, since VPN sites belonging to different VPNs can have overlapping IP addresses, it is not possible to use the site IP alone as a BGP prefix in the NLRI field. It is necessary therefore to distinguish the routes. This is why each VRF has by configuration a route distinguisher. Each site IP address is added the route distinguisher of the VRF that connects it. The concatenation $\langle Route\ distinguisher : Site\ IP \rangle$ forms a BGP prefix in the NLRI field that is unique in the network.

BGP is a policy based protocol, it has attributes that can specify how to treat BGP routes. The extended community attribute, carries the route target attribute (RT). The RT specifies where the route (mapping) will be ultimately installed. It is therefore possible, thanks to the RT, to exchange mappings **only** between VRFs of the same VPN that need to interconnect. In practice, VPN customer topology (Hub and spoke, full mesh of sites, etc...) is expressed by the customer when it contracts the VPN service. The provider configures therefore its VRFs according to this topology with appropriate route targets. In effect, each VRF has by configuration a set of RTs that it imports and a set of RTs that it exports. When a PE announces a route contained in a VRF, it tags it with the export RT(s) of this VRF. When a destination PE receives the BGP route, it looks to the RT(s) and checks if one of its VRFs imports it. If one VRF is configured with the same import RT, the PE installs the route in this VRF. Therefore, in order to interconnect two VRFs and make them part of the same VPN, the provider backbone needs only to configure the two VRFs to import and export the same RTs.

C. Mapping distribution in a large BGP network

Starting from a few tens of PE routers, instead of having a full mesh of BGP sessions between PE routers which does not scale (See Sec. IV-B), provider backbones often use a hierarchy of route reflectors (RRs). Route reflectors (RRs), represented with the RR diamonds in Fig. 3, are BGP routers that establish BGP sessions with PEs. Their role is to help the PEs exchange BGP routes between each other. They usually do not participate in MPLS forwarding of VPN traffic.

IV. WHY BGP IS NOT THE BEST FIT FOR VPN ROUTING

Starting from the basic routing needs described in Sec. II-C, this section discusses the choice of BGP as the signaling protocol to provide VPN routing. We first draw up the evolution of BGP MPLS deployment and show its limitations.

A. History of BGP MPLS IP VPNs

One of the first studies on BGP MPLS IP VPNs is a formal analysis of the RFC describing this technology. The study compares the desired service model to the forwarding model and describes conditions that need to be satisfied in order to provide the necessary isolation between different VPNs [12].

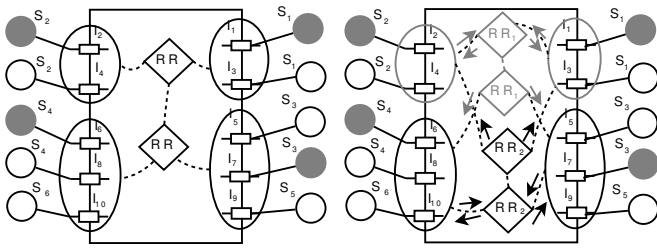


Fig. 3. Route reflection (left) and multiple plane RR (right)

With the first deployments of this technology came the first measurement studies. Pei et al. [8] focused on the BGP convergence in such networks. Their study was based on both testbed experiments and passive measurements on data collected from a large VPN provider backbone. Their work gives an evidence that BGP convergence problems due to path exploration and route invisibility are not exclusive to Internet backbones.

Another study made by a large VPN provider [7] reported a route explosion in such networks. The large provider reported that the number of its BGP routes went from 400,000 in 2005 to 1,800,000 in 2008. This route explosion is due to the high number of customers that such networks can support. This route explosion has a direct impact on operational networks since it affects the scalability of RR routers that have to maintain all these routes. Another consequence of route explosion was revealed by a measurement study on BGP routing table transfers [6]. The study observed that table transfers in a large VPN provider backbone can take several minutes when a high number of routes needs to be exchanged.

In order to tackle the scalability problem of RR routers, one large VPN provider uses VPN partitioning or multiple plane RR design [7]. The multiple plane RR solution consists in having two, or more, separate BGP RR control planes, each handling routes from a set of PE routers. Fig. 3 (right) shows an example of a VPN provider network with four PEs and two RR planes, the gray and the black. Each of the RR planes receives routes from two PEs and distributes routes to the four PEs. In a large provider backbone, by carefully assigning the PEs to the RR planes, having two control planes for instance, results in having twice fewer routes on the RRs of each of the RR planes. Unfortunately, it also results in having twice as many RRs which increases the costs and the configuration complexity. Setting the cost aside, this solution is hard to evolve and manage as the distribution of VPN customers among PE routers changes across time. This can result in unbalanced RR partitions that might need to be re-architected.

In order to reduce the size of RR routing tables as well as the amount of BGP messages exchanged, the IETF proposed “constrained route distribution” [9] (also known as RT-constraints). With the basic BGP MPLS standard [3], routes are flooded to all the BGP routers in the network and routers apply import policies to select the routes they need for their VPNs and discard the rest. RT-constraints is an outbound filtering based

mechanism in which BGP speakers exchange RT membership information. This mechanism allows each router to pre-filter routes before it sends them to its BGP neighbors resulting in fewer messages and a better distribution of routes. The RT-constraints mechanism allows RRs for instance to pre-filter the BGP routes and send only the routes that PEs will ultimately install. Unfortunately, this solution has two limits. First, it complicates the task of route reflectors which were designed to simply pack updates and send them. Instead, each RR would need to keep an outbound filter per neighboring PE. Second and most importantly, the fact that it reduces the RR routing table size is very arguable; RR routers still have to keep routes of all VPNs that connect to their client PE routers (See Sec. VI-C1 for an evaluation of RT-constraints).

As for the slow routing table transfers, Houidi et al [6] show that this slowness is mainly due to the timer-driven implementation of the interaction between BGP and TCP. The study shows that an event-driven implementation increases the table transfer speed by one order of magnitude but only for transfers that affect a unique BGP session. However, the study shows that transfers spanning multiple BGP sessions remain slow when a high number of routes need to be exchanged. As VPN RRs have both large routing tables and many BGP sessions to PE routers, table transfers from one RR to its client PE routers remain an issue.

B. BGP with Hop-by-Hop signaling is to blame

A routing protocol is a signaling protocol in which signaling nodes exchange state information such as routes or links in order to build routing tables. Signaling protocols in general can be classified into hard state, soft state or a mixture of both [14]. When applied to multi-hop systems, they can be implemented Hop-by-Hop or End-to-End [14], [15]. In End-to-End signaling, state is exchanged exclusively between state source and state destination nodes. In Hop-by-Hop signaling, intermediate nodes participate in the signaling process by relaying state messages.

Following the above classification, BGP signaling within a single autonomous system (iBGP) can be either Hop-by-Hop with route reflection or End-to-End with a full mesh of iBGP sessions. Since each BGP router is a state sender, a state holder and a state receiver at the same time, using a Hop-by-Hop signaling implies that intermediate routers keep the signaling state for all the routers in their signaling system. This is what route reflectors do when used in VPN routing. Sec. IV-A shows that such a Hop-by-Hop approach is not sustainable starting from a large number of BGP routes.

In fact, Hop-by-Hop signaling can be suitable when all routers need to keep the “full state”. This is the case for Internet routing, where BGP nodes usually need all routes to realize full Internet connectivity. However, as seen in Sec. II-C, in VPN routing, each signaling node needs to handle only a part of the state, namely the state needed for VPNs that are connected to it. Moreover, unlike Internet backbones where BGP routes can be aggregated, VPN customer routes can not be aggregated because of control plane isolation requirements.

Solutions	State in the core	Configuration overhead	# sessions	Cost
RR	×	✓	✓	≈
multiple RR	≈	≈	✓	×
RR + RT	×	✓	✓	≈
Full mesh	✓	×	×	✓
Full mesh + RT	✓	×	×	✓
What we need	✓	✓	✓	✓

TABLE I
QUALITATIVE COMPARISON OF EXISTING VPN ROUTING APPROACHES

Indeed, customers are free to choose their addressing and providers have no control over it.

It is clear therefore that an End-to-End signaling approach is the only way to remove the state from the core of the network. This approach has, however, the drawback of PE routers still receiving all the routes in the network whether they need them or not. This approach can be therefore coupled with an outbound filtering mechanism like RT-constraints [9] to optimize the route distribution inside the provider backbone. Unfortunately, although it removes the state from core network, even these two coupled solutions are not convenient. In practice, a full mesh of BGP sessions becomes overwhelming starting from few tens of routers. Matter of fact, this solution has the following scalability concerns when applied to VPN routing [16]:

- Configuration overhead: Each time a BGP neighbor is added, all the other routers in the network must be reconfigured.
- The burden of maintaining extreme number of BGP sessions, sometimes with different policies.

Following the above analysis, (1) BGP Hop-by-Hop signaling with route reflection results in "full state" in the core which is not sustainable, while (2) a full mesh of BGP sessions is not scalable, hard to administrate and evolve. At least two "add-on" solutions, RT-constraints and multiple plane RR, can be coupled with solutions (1) or (2) to enhance BGP VPN routing. Unfortunately, these add-on solutions are either inefficient or short term. Obviously, there is a need for a VPN routing solution with the following properties: no full state in the core, easy to configure and evolve, with no heavy burden on end routers. Table I summarizes current solutions advantages and drawbacks, with the sign × meaning that the solution does not satisfy the desired property, ✓ that it satisfies it and ≈ meaning that although the solution does a good job satisfying the property, it can be improved.

In this paper, we focus on the scalability of the VPN route distribution. It is important to note that there are other aspects of VPN routing scalability that are out of the scope of this paper. One of these is the size of PEs forwarding tables. Relaying [17], for instance, is an indirection-based solution that aims at reducing the size of PEs forwarding tables at the cost of having extra traffic and delay. Since this solution does not change the way VPN routes are distributed, it can as such be coupled with our solution that we present next.

V. TWO-STEP VPN ROUTING

We now present Two-step VPN routing, an End-to-End approach for a more scalable VPN routing.

A. Intuition behind Two-step VPN routing

When BGP is used for VPN routing, the state information involved in the signaling between BGP routers can be summarized into the following:

- (1) *The state of BGP routes* (the route and whether it is up or down).
- (2) *The fact that the BGP node is in the signaling system* (thanks to BGP sessions).
- (3) *Information about how BGP routes should be treated and where they should be installed* (thanks to BGP attributes).

Each of these state information express one functionality necessary for VPN routing. State information (1) informs about VPN site *reachability*, while state information (3) informs about the *topology* of VPN sites.

VPN site reachability informs both on whether VPN sites are reachable or not and on how to reach them. This is inferred at the level of a PE router, for instance, thanks to the routing protocol that runs between the PE router and the customer router that connects the VPN site. By *VPN topology*, we mean the way the sites of a VPN must be connected to each other (full mesh, overlapping VPNs, hub and spoke etc). VPN topology is expressed in the BGP MPLS standard using the route target attribute. Finally, state information (2) keeps track of the nodes in the VPN routing signaling system. In the case of BGP, this is done thanks to BGP sessions: as long as the BGP session is up, the BGP node is in the signaling system.

BGP is not scalable for VPN routing because it couples these three distinct functionalities ((1),(2) and (3)) into one unique entity that is BGP. BGP with Hop-by-Hop signaling (RR) scales the second functionality because there are less sessions to maintain but causes scalability concerns with the first functionality because RRs must maintain state for all the routes. On the other hand, BGP with End-to-End signaling scales the first functionality (No state for routes in the core), but causes scalability concerns with the second functionality because each PE needs to maintain a BGP session with each of its neighbors. In both cases, Hop-by-Hop or End-to-End, the third functionality causes scalability concerns because BGP routes are always flooded. It is up to end routers to pick the routes they need for their VPNs. An outbound filtering mechanism like RT-constraints can fix this problem of flooding of routes but it causes each BGP router to keep an outbound filter for each of its neighbors.

In order to scale VPN routing, we adopt an End-to-End signaling approach and propose to separate these three functionalities. End-to-End plus the separation of the three functionalities are the basic ideas behind the Two-step VPN routing approach. This approach consists in separating VPN topology from VPN site reachability. In a first step, PE routers exchange VPN topology information. At the end of this step,

each PE router is VPN topology aware: it is able to know for each of its VPN interfaces, the set of destinations that are interested in receiving mapping information (routes) for this interface. This first step is also responsible for functionality (2), that is, knowing which PE routers are part of the VPN routing signaling system. Then, in a second step, based on the output of the first step, each PE router sends its VPN site reachability information (mapping) only to destinations that need it. We first detail the first step of our VPN routing approach.

B. Step 1: Building VPN topology

1) *Overview*: This step builds and updates a *VPN adjacency table* for each VPN interface. The VPN adjacency table of a given interface contains the set of destination PEs that need to receive mapping (reachability) information from this interface.

Our approach does not change the way a VPN operator provisions its VPN customers. VPN topology is still expressed by VPN customers and is expressed by the VPN operator into the configurations of PE routers. Each VPN interface has by configuration a set of RTs that it imports and a set of RTs that it exports. A VPN interface I_1 that imports RT_1 accepts to receive mapping information from any interface I_i that is configured to export RT_1 .

In order to build the VPN adjacency table, each PE router needs to flood the following information:

- The PE address
- The set of RTs that the VPN interfaces of the PE import.

Each PE needs also to store the VPN topology information that it receives and use it to populate its VPN adjacency tables whenever there is a change on its exported RTs. When a PE destination router receives topology information, say from PE_1 , it looks if one of its VPN interfaces is configured to export one of the RTs contained in the topology information. If it is the case, the address of PE_1 is added to the VPN adjacency table of the interface that exports the considered RT. Next, we explore different ways to implement this step.

2) *Implementing the VPN topology step*: The VPN topology step has to perform two tasks: update topology information and know which PEs are in the VPN signaling system and whether they are up or not. This step is performed by what we call the *VPN topology tracker* in Fig. 4. The topology tracker gets topology information from the configurations, forms topology information messages and floods them to all the PEs in the network. Upon the reception of a topology message, the topology tracker checks whether there is a change in the topology and updates the corresponding VPN adjacency tables. We see mainly two ways of implementing the topology tracker tasks, either hard state or soft state.

A possible hard state implementation can use a BGP-like mechanism in which PE routers form a BGP signaling overlay in order to exchange VPN topology information. In this case, detecting that a PE router is in the signaling system is done hard state thanks to the BGP sessions between PE routers. The messages can be encoded as the following. The PE IP address can go into a special NLRI field (possibly a new address

family) and the set of RTs imported by the VPN interfaces can go into one of the BGP attributes field, possibly the Extend communities attribute. Each change in the configuration of RTs on VPN interfaces triggers a BGP message that updates the previous topology information. This implementation has the advantage of natively supporting inter domain VPNs and being easily implementable on the top of current software. Such an implementation has however the drawbacks of BGP in terms of the number of sessions per PE. In this particular case, the use of route reflection is not blocking since RRs would need to keep only few routes, namely one BGP route per PE router (therefore few hundreds of routes).

A soft state approach can rely on the IGP that is already running in the VPN backbone. VPN topology information could be then flooded in special TLVs [18] together with ISIS or OSPF periodic update messages. The drawback of this approach is that inter domain VPN routing has to be thought separately. Typically, if a VPN spans two ASs for instance, this information needs to be redistributed across ASes.

C. Step 2: Exchanging reachability information

Once the VPN topology is set up, reachability information can be sent separately using the information contained in the VPN adjacency table.

1) *Overview*: A change in the VPN site reachability is detected thanks to the routing protocol that runs between the VPN site and the PE interface that connects it (PE-site1 routing protocol in Fig. 4). This change is first updated on the VPN routing information base (RIB) as well as on its forwarding information base (FIB). This change needs then to be propagated to distant PE routers that are concerned. This is the role of the second step in our approach. This step is performed by what we call the *VPN interfaces manager* in Fig. 4. Upon the detection of a change that needs to be propagated, the VPN interfaces manager creates a message containing the change and sends it to all the PEs that are in the VPN adjacency table. The message contains the following information:

- (Site id,Interface locator) mapping
- The RT(s) in the export list of the VPN interface (needed to know which VPN interface needs the mapping)
- Possibly the nature of the change: route up/down (depending on whether the approach is hard or soft state)
- Attributes for routing policies²

The other way around, when a VPN interfaces manager receives a route, it first looks at the RT contained in the route in order to know which VPN interface is concerned by the route. Then, it updates the VPN RIB of the corresponding VPN interface.

2) *Implementing the VPN reachability step*: The VPN reachability step takes as an input the VPN adjacency table built by the VPN topology tracker in the first step. Whenever

²We do not detail the use of routing policies because it is orthogonal to our approach. Implementing this feature can be done exactly as it is done with BGP by assigning “local preferences” to mappings.

1) *Input VPN routing data:* We collect routing data from a large VPN provider that has hundreds of PE routers to host hundreds of thousands of VPN sites. At the time of the data collection, the provider backbone uses clusters of route reflectors for VPN site route distribution with no multiple plane RR and no RT-constraints. Each set of PE routers is a BGP client of a cluster of route reflectors. Each cluster has two route reflectors for redundancy. Following this configuration, each PE has by configuration a BGP session to the two route reflectors of its cluster. Clusters of route reflectors are fully meshed with BGP sessions to allow BGP route distribution between all the clusters.

We use two kinds of input data from the network:

- **BGP update messages:** A trace collector is configured as a BGP client of route reflectors. It collects the BGP routing announcements flooded over the networks. We collect two months of BGP routing updates.
- **Router configurations:** We obtain a snapshot of router configurations from all PE routers and route reflectors. The snapshot is very close in time to the boot of the BGP sessions between our collector and route reflectors. We develop tools to automatically parse router configurations, which takes into consideration the different router configuration languages.

We process router configurations to infer the VPN topologies and create an instance of our model presented in Section II. We process the updates to observe the dynamics of the routing data exchanged for the input model created using the router configurations. Note that using the routing updates alone does not allow us to infer VPN topologies since BGP routes contain only exported RTs.

2) *Building the input provider network model:* Thanks to router configurations, we extract for each PE router its IP address and the set of VPN interfaces (VRFs in BGP MPLS VPN terminology) that it hosts. For each of these VPN interfaces we extract the set of imported Route Targets (RT), the set of exported RTs and the route-distinguisher (RD) attribute. We discard RTs on the router configurations corresponding to administration or testing.

We use the set of imported RTs and exported RTs to build VPN interface adjacencies. Two VPN interfaces are adjacent if and only if they import and export each others RTs. Proceeding this way, we build the VPN interfaces graph in which we have a vertex per VPN interface and an edge between two VPN interfaces if the VPN interfaces are adjacent. Given the graph of all VPN interfaces, a VPN is a connected component in this graph. This definition is powerful because it allow us to build VPN topologies only from router configurations without any prior assumptions on the topology or the naming conventions of the usage of RTs.

Processing router configurations as previously explained, we obtain an input provider model that we use to evaluate our solution. The model has more than 10,000 VPN topologies and almost 100,000 VPN interfaces. Our results therefore apply on this studied provider backbone. More analysis on a wider range of VPN topologies can reinforce our conclusions. We

leave such analysis for future work, the goal of this paper being to demonstrate the need for a dedicated VPN routing protocol and to show the potential of Two-step VPN routing.

3) *Emulation of mapping information changes:* BGP routes contain information that allow us to identify the VPN interface that originated each route. Identifying an interface is done thanks to Next Hop attribute that contains the PE IP address and to the RD in the NLRI field that identifies the VRF that originated the route(See Sec. III-B for more information). Using the input model that we built in the previous section, we are able to transform BGP routing update messages into their equivalent with a Two-step VPN routing approach. That is, a BGP route is transformed to a (site id, interface locator) mapping originated by a given VPN interface (PE IP address, RD). In Two-step VPN routing words, the network input model allows us to build the VPN adjacency tables for all VPN interfaces and BGP routing updates gives us a realistic input for the VPN sites reachability changes.

Since we are only interested in routing updates generated by PE routers, we pre-process routing updates to remove updates generated by session resets either between the RR and the trace collector or between RRs. Since we have a unique snapshot of router configurations, we also pre-process routing updates to discard those that do not correspond to VPN interfaces that we inferred from the configurations. These updates that we discard correspond to either inter domain VPN routes or new VPN interfaces that has been configured after we extracted the configuration snapshot.

B. Design choices of the two-step VPN routing approach

This section validates our design choices using the input model that we built from the provider backbone.

1) *PE routers do not need "full state":* By coupling router configurations and routing data we are able to quantify our assumption about PEs not needing the "full state" in order to realize VPN routing. The number of mappings a PE needs is the sum of the number of mappings originated by distant VPN interfaces that are in its adjacency table. We infer this number for each PE from an RR's routing table since it contains the "full state". The curve 'Mappings needed' in Fig. 5 shows the cumulative percentage of mappings needed by each PE router. The figure validates our assumption by showing that in 70% of the cases, a PE router needs less than 10% of all mappings in the network. More, half of the PEs do not need to receive more than 5% of the total mappings in the network in order to realize VPN routing.

2) *VPN topology is stable compared to VPN reachability:* Because we have only a snapshot of the router configurations before the month of data collection, we could not check for all the modifications of VPN topologies. We are able, however, through routing updates to detect new VPN interfaces that have been added to the network by detecting new RDs that were not in the input model. The routing updates naturally allow us to see the dynamics of VPN site reachability. We compute the occurrence of mapping changes and those of topology changes

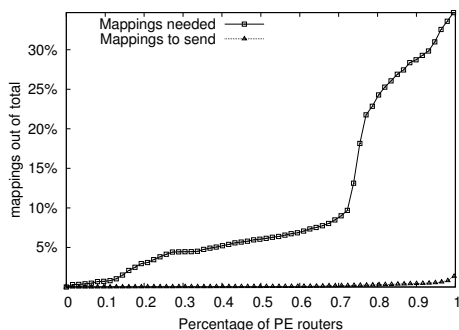


Fig. 5. Mappings announced/needed by PE

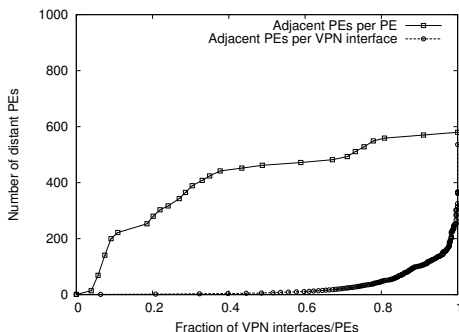


Fig. 6. VPN adjacency tables

during one month and find that reachability changes are three orders of magnitude more frequent than topology ones.

3) *There is a logical full mesh of PE routers:* It is tempting to take an End-to-End approach with BGP while at the same time avoiding full mesh of BGP sessions. The temptation comes from the fact that only PEs that share the same VPN customers need to exchange routing information. Studying the VPN topology of our input model, we find that although 95% of VPN topologies span less than 17 PE routers, there is a small fraction of large VPN topologies that are located in many PEs. These large VPN topologies make that PEs are highly meshed as we can see in the 'Adjacent PEs per PE' curve in Fig. 6: 70% of the PEs need to talk to at least 400 distant PEs. We compute that such an End-to-End signaling approach with BGP needs at least 66% of the $\frac{N(N-1)}{2}$ links in the full mesh case.

C. Comparison of VPN routing approaches

We compare the solutions discussed in Sec. IV-B in terms of the size of routing tables in the core (state to maintain in the core), the number of messages generated by mapping information changes and PE routers activity.

1) *State in the core in Hop-by-Hop approaches:* There are two solutions to reduce the state in the core in hop-by-hop signaling with route-reflection, multiple plane RR and RT-constraints. The gain of multiple plane RRs is straightforward: a two plane RR design reduces the RRs routing table size by two at the cost of having twice as many RRs. Quantifying the gain of RT-constraints is less trivial since each RR cluster needs to keep routes originated by its PE client routers plus

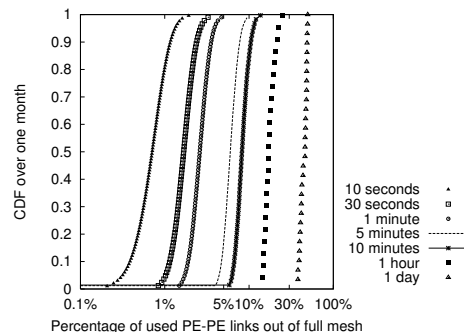


Fig. 7. PEs involved in signaling per slot of time (1 month)

the number of routes originated by distant VPN interfaces and that are needed by its PE client routers. Thanks to our methodology, we compute the gains of RT-constraints for each of the clusters of the studied provider backbone. We find that RT-constraints is surprisingly not enough efficient. At best, it removes around 16,6% of the routes maintained by an RR. With one of the clusters, the gain is even less that 6,5%.

2) *The burden on PE routers with End-to-End approaches:* Removing the state from the core has a cost for End-to-End signaling approaches. Whereas in RR solutions, a PE needs to send its routes only to few route reflectors, in End-to-End approaches, PE routers exchange mapping information directly between each other.

In full mesh of BGP sessions (with or without RT-constraints), a PE router has to maintain BGP sessions with all its neighbors and also to update all its PE neighbors. With a Two-step VPN routing, a PE router only has to send mappings when necessary. If we set aside the configuration complexity of full mesh BGP routing, then one might think that if all PEs permanently communicate routing information with each other, Two-step VPN routing could be considered equivalent to a full mesh of BGP routers. We study a period of one month of reachability information changes to extract for each time slot, the set of PEs that participate in signaling. We find that each slot of 5 minutes, less than 5% of the $N(N-1)/2$ total possible PE-PE links in a full mesh participate in signaling. Fig. 7 shows the percentage of PE-PE links used to transport mapping information changes during various slots of time. The figure shows that even if we consider a period of one day then only between 30% and 40% of the total PE-PE links are concerned by reachability changes. There is therefore clearly a gain in sending routes only when needed and not maintaining permanent BGP sessions like it is the case with BGP full mesh and BGP full mesh + RT-constraints.

Now, with Two-step VPN routing, a mapping change still has to be sent to all the PEs in the site's VPN interface adjacency table which might be an issue. The curve "adjacent PEs per VPN interface" in Fig 6 shows the cumulative number of distant PEs to update for VPN interfaces. The figure shows that fortunately, for 50% of the VPN interfaces, a mapping change needs to be propagated to less than 10 distant PE routers. Moreover, for 92% of VPN interfaces, a change needs

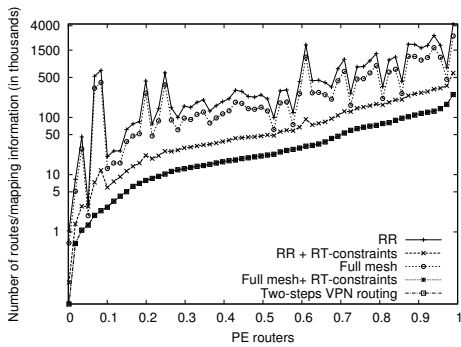


Fig. 8. Comparison of different approaches at PE router boot

to be propagated to less than 130 distant PEs, which is the number of PE clients that a route reflector needs to update in the studied provider backbone. Nevertheless, few VPN interfaces need to update almost all the PE routers in the network when they see a mapping change. Also, when a PE router boots, it needs to update all its adjacent PEs which can reach as well all the PEs in the network (See 'Adjacent PEs per PE' curve in Fig 6). We argue that this is not an issue considering the small size of routing data that needs to be sent by PE routers. The curve 'Number of mappings to send' in Fig. 5 represents the number of distinct mappings that a PE router sends when it boots: almost all PE routers generate each less than 1% of the total number of mappings in the network. A PE router boot is therefore an easy task compared to what an RR does when it sends its entire routing table (100 times bigger than that of PE) to the hundreds of PE routers that are its clients. We next consider this worst case scenario of a PE router boot and compare the number of messages generated after such an event for each of the existing approaches.

3) *Number of messages generated in the network:* With both Two-step VPN routing and BGP, mappings (reachability information) are duplicated in order to be distributed. We evaluate on our input model the number of mappings generated by different approaches after the boot of a PE router.

For each solution we count the number of mappings sent including duplicates of the same mapping. That is, when an RR sends the same mapping to 100 PEs we count it as 100 mappings sent. Fig. 8 shows the number of mapping information generated by each PE router when it boots. The y-axis (in log scale) represents the number of mapping information while the x-axis presents the PE routers sorted by the number of mappings generated in a Two-step VPN routing. The figure clearly shows that Two-step VPN routing and Full mesh iBGP + RT constraints have similar performances and generate much less routing data. The RR solution together with full mesh are the worst because messages are flooded to all the PEs whether they need them or not. RR generates more messages than Full mesh because of both redundancy inside clusters and the fact that the messages are sent 'hop-by-hop' between RRs.

VII. CONCLUSION

This paper shows that BGP is not appropriate for large scale VPN routing because it causes some routers to keep

state for all routes in the network. We propose Two-step VPN routing, an End-to-End signaling approach in which we separate two concerns: the creation of VPN topology and the advertisement of VPN site reachability. End-to-End signaling guarantees that no router will maintain the full state and that no path exploration will happen; the separation of topology and reachability allow us to scale the End-to-End signaling. We couple router configurations and routing updates collected from a large VPN provider to validate our design choices and predict the behavior of our approach. We show that our solution outperforms state-of-the-art proposals in reducing routing table size and generating less routing messages. End-to-End signaling with BGP is theoretically possible and capable of removing the state from the core. However, it is hard to configure and causes PE routers to maintain permanent sessions between each other. Our study of End-to-End routing activity shows that permanent sessions are not required to perform signaling under typical network conditions.

ACKNOWLEDGMENTS

We thank our shepherd, Matthew Caesar, for his useful feedback. We are grateful to Renata Teixeira for her support and feedback on this work since its earlier versions. We also thank Bruno Decraene for his valuable comments.

REFERENCES

- [1] D. Fedyk, Y. Rekhter, D. Papadimitriou, R. Rabbat, and L. Berger, "Layer 1 VPN Basic Mode." RFC 5251, Jul 2008.
- [2] L. Andersson and E. Rosen, "Framework for Layer 2 Virtual Private Networks (L2VPNs)." RFC 4664, Jul 2008.
- [3] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks." RFC 4364, Feb 2006.
- [4] K. Kompella and Y. Rekhter, "Virtual Private LAN Service (VPLS) using BGP for Auto-Discovery and Signaling." RFC 4761, Jan 2007.
- [5] P. Knight, H. Ould-Brahim, and B. Gleeson, "Network based IP VPN Architecture Using Virtual Routers." Internet-Draft, Mar 2006.
- [6] Z. B. Houidi, M. Meulle, and R. Teixeira, "Understanding slow bgp routing table transfers," in *Proc. Internet Measurement Conference*, 2009.
- [7] M. Napierala, "AT&T MPLS Network and VPN Services." PLNOG, 2008.
- [8] D. Pei and J. V. der Merwe, "BGP Convergence in Virtual Private Networks," in *Proceedings of the 6th ACM SIGCOMM on Internet measurement*, 2006.
- [9] P. Marques *et al.*, "Constrained Route Distribution for BGP/MPLS IP VPNs." RFC 4684, Nov 2006.
- [10] R. Hinden, "New Scheme for Internet Routing and Addressing." RFC 1955, 1996.
- [11] S. Deering, "The Map & Encap Scheme for Scalable IPv4 routing with Portable Site Prefixes." Presentation, Xerox PARC., Mar 1996.
- [12] R. Bush and T. Griffin, "Integrity for Virtual Private Routed Networks," in *Proceedings of the IEEE INFOCOM*, Apr 2003.
- [13] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol extensions for BGP-4." RFC 4760, 2007.
- [14] P. Ji, Z. Ge, J. Kurose, and D. F. Towsley, "A comparison of hard-state and soft-state signaling protocols," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 281–294, 2007.
- [15] X. Fu and J. He, "End-to-end versus hop-by-hop soft state refresh for multi-hop signaling systems," in *ICNP*, pp. 171–180, 2009.
- [16] D. Walton, "BGP Scalability and Troubleshooting." RIPE 42 meeting <http://www.ripe.net/ripe/meetings/ripe-42/index.html>, Apr 2002.
- [17] C. Kim, A. Gerber, C. Lund, D. Pei, and S. Sen, "Scalable VPN Routing via Relaying," in *Proc. ACM SIGMETRICS*, 2008.
- [18] D. Oran, "OSI Intermediate System to Intermediate System Routing Protocol," in *RFC*, Feb 1990.