A Cooperative Approach to Interdomain Traffic Engineering

Bruno Quoitin and Olivier Bonaventure
Department of Computer Science and Engineering
Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium
Email : (quoitin,bonaventure)@info.ucl.ac.be

***Abstract*— For performance or cost reasons, Autonomous Systems (AS) often need to control the flow of their incoming interdomain traffic. Controlling its incoming traffic is a difficult task since it often implies influencing ASes on the path. The current BGP-based techniques that an AS can use for this purpose are primitive. Moreover, their effect is often difficult to predict.**

In this paper, we propose to solve this problem by using *Virtual Peerings*. A *Virtual Peering* is an IP tunnel between a border router of a source AS and a border router of a destination AS. This tunnel is established upon request from the destination AS. These tunnels can be negociated by using backward compatible modifications to the Border Gateway Protocol (BGP)

By using *Virtual Peerings*, the source and destination ASes can achieve various traffic engineering objectives such as traffic-balancing or reducing the latency. A key advantage of our solution is that it does not require cooperation of the intermediate ASes and that it can be incrementally deployed in today's Internet. We then show by simulations that in a load-balancing scenario, a multi-homed AS only needs to request a few dozens of *Virtual Peerings* to balance its incoming traffic.

I. Introduction

The Internet was created as a best effort experimental network and was mainly used by researchers and students. During the last years, it has evolved to become a major worldwide network that is used to carry various types of mission critical traffic. The Internet is divided in Autonomous Systems (ASes) that exchange routing information with the Border Gateway Protocol (BGP) [1], [2]. As of this writing, there are about 17,000 ASes. About 12,000 of those ASes are stub ASes and the remainder are transit ASes. A stub AS is an AS that does not allow transit traffic to cross its network.

Given the cost and importance of their interdomain traffic, ASes often need to optimise the flow of their interdomain traffic [3], either for performance reasons (e.g. using low delay paths) or cost reasons (e.g. using the cheapest provider). Content providers are net producers of packets and thus they need to control the flow of their outgoing packets. For this, they can tune the configuration of their border routers to fully control the selection of their outbound routes [4], [5]. Access providers such as Asynchronous Digital Line Subscriber (ADSL) or Cable TV (CATV) providers or campus and enterprise networks are packet consumers. Thus, they often need to control the flow of their incoming traffic [4], [3]. Unfortunately, controlling the incoming traffic is more complex as the path followed by the incoming packets depends on the outcome of the BGP decision process on all the transit routers between each source and the destination AS. The current techniques used by such ASes to control their incoming traffic, namely as AS-Path prepending, selective announcements, the advertisement of more specific prefixes or the use of BGP communities [4] suffer from several drawbacks. They are often coarse grained and it is difficult for an AS to predict the impact of one BGP change to its incoming traffic [6], [4], [7]. Some of those hacks increase the size of the BGP routing tables.

In this paper, we propose a cooperative approach to allow an AS to control its incoming traffic in a deterministic manner. Our approach relies on the establishment of *Virtual Peerings*. A *Virtual Peering* allows a destination AS to request a source AS to send its packets via a chosen ingress router in the destination AS. Our solution can be used by ASes willing to load-balance their incoming traffic or use low-latency or high-bandwidth paths. We focus on stub ASes such as content-providers, enterprise networks and broadband access providers that produce and sink most of the traffic in the Internet. Though our solution can also be used in the case of transit ASes, controlling the incoming traffic in large transit ASes is a different problem that is outside the scope of this paper.

Our solution slightly modifies the BGP protocol. To ensure that those modifications can be deployed incrementally, we do not require transit ASes to support our extensions. The only affected routers are located within the cooperating source and destination. This is a key contribution of our solution.

The remainder of this paper is organised as follows. In section II, we briefly describe our proposed architecture. In section III, we detail the establishment of the *Virtual Peerings* and the required modifications to BGP. We present one possible applications of Virtual Peerings in section IV. Then, we evaluate the performance of our scheme by considering traffic-balancing as one possible traffic engineering objective. Section V shows, by means of simulations that it is possible for a stub AS to load-balance its incoming traffic by establishing only a fen tens of *Virtual Peerings*. Finally, we compare our approach with related work in section VI.

II. Virtual peerings

Today, a common method used by ASes to engineer the flow of their interdomain traffic is to establish peerings with other ASes [8]. Those peerings are established either through direct private links between the two ASes or over an interconnection

point. An eBGP session is used over the peering link to advertise the prefixes that are reachable via each AS. BGP peerings are established manually by changing the routers configurations by hand. However, manual operations are error-prone and slow. In addition, the time of establishment of a new peering is often on the order of magnitude of several days or weeks.

We propose to solve these problems with Virtual Peerings, which automate the establishment of BGP peerings between cooperating ASes and extend them to non-adjacent ASes. Virtual Peerings allow an AS to control the ingress point used by a non-adjacent AS. Therefore, Virtual Peerings represent a deterministic solution to the control of an AS's incoming traffic. Virtual Peerings are established by cooperating ASes based on the current traffic load or another property. We expect that the establishments and removals of Virtual Peerings will occur on a timescale of at least a few hours.

A *Virtual Peering* is a peering built on a dynamically established uni-directionnal IP tunnel between two cooperating, but non-adjacent, ASes. This tunnel is used by the source AS to send packets to the destination AS via a chosen ingress router in the destination AS. For this purpose, we propose to place, inside each cooperating AS, a *Virtual Peering Controller* (VPC) that will be responsible for the establishment and maintenance of Virtual Peerings. A VPC can for example be a dedicated workstation or a stand-alone BGP router.

Various types of IP tunnels can be used to carry the packets on the Virtual Peerings. The simplest solution is to use IP-in-IP encapsulation or Generic Routing Encapsulation (GRE) tunnels. Those solutions have a low overhead (20 bytes for IP-in-IP and 24 bytes for GRE) and are supported by most routers. Two other possible types of tunnels are Layer Two Tunneling Protocol (L2TP) and IPSec in tunnel mode. L2TP is often used to provide Virtual Private Network (VPN) services, but its overhead is larger than GRE. The main advantage of IPSec would be its authentication and encryption facilities that could be used to protect the Virtual Peering.

In the past, IP tunnels have often been criticised because of the cost of encapsulating/decapsulating packets and the risk of fragmentation. We would like to point out that those are not operational problems anymore. High-end routers are now capable of supporting line rate encapsulation and decapsulation, either on the normal interfaces or by using special interfaces [9]. Second, with Packet over SONET/SDH links that are widely deployed by ISPs, the MTU is less stringent as it was earlier. Furthermore, almost all TCP implementations support PathMTU discovery and the tunnel head-end could also perform PathMTU discovery on the tunnel itself.

Another common type of tunnels used by ISPs are MPLS tunnels. For Virtual Peerings, MPLS would offer a lower overhead as well as fast restoration, bandwidth reservation and traffic engineering capabilities. Unfortunately, those advantages come at a price: the transit domains must support MPLS and must allow other domains to use RSVP-TE to establish MPLS tunnels through their own network. While many large ISPs use MPLS inside their network , they are often reluctant to let their customers or peers send RSVP-TE messages to establish MPLS LPSs through their network.

To understand the operation and the usefulness of Virtual Peerings, let us consider the simple network shown in Fig. 1. In this network, assume that ASD would like to balance over its providers the traffic received from the two sources AS1 and AS2. As the two sources are attached to the same provider, neither AS-Path prepending nor redistribution communities [7] would allow ASD to control its incoming traffic.

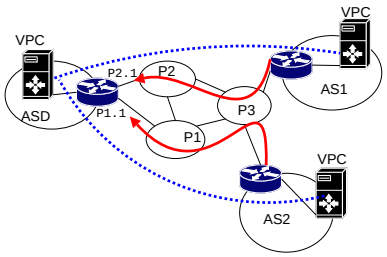

Fig. 1. Sketch of the approach

As the access router of ASD is attached to two providers, P1 and P2, another solution is possible. When a provider establishes a link with one of its customers, it usually allocates two IP addresses on this link from one of its own CIDR blocks. The first one is for its own router and the second one is for the router on the customer side. A consequence of this common practice is that the access router of ASD can be reached via two distinct IP addresses: P1.1 and P2.1. As P1.1 belongs to the CIDR block advertised by P1, any packet sent on the Internet with P1.1 as destination will reach ASD via P1. Based on this finding, ASD can balance its incoming traffic provided that it can convince AS1 (resp. AS2) to send all the packets whose destination belongs to ASD inside an IP tunnel that terminates at P2.1 (resp. P1.1). This tunnel can be established without any cooperation from the transit providers. It is entirely controlled by AS1 (resp. AS2) and ASD.

III. Architecture and protocol

Multiple components are involved in the establishment and operation of a Virtual Peering, as shown in Fig. 2. First, the two autonomous systems that will establish the Virtual Peering: the Requester AS (RAS) and the Source AS (SAS). The RAS is the AS that is willing to control its incoming traffic. One of its router will terminate the tunnel that will originate in one router of the SAS. Both the RAS and the SAS can be networks composed of several BGP routers. There must be at least one Virtual Peering Controller (VPC) in each SAS and RAS. The VPCs are responsible for monitoring the network and establishing the required Virtual Peerings. In order to monitor the traffic, VPCs are linked to a measurement infrastructure such as [10] In addition, VPCs will have an iBGP session with all the routers in their domain. VPCs can be dedicated workstations or stand alone BGP routers. Due to their central position in an AS, it would be natural to implement the VPC features on BGP route-reflectors.

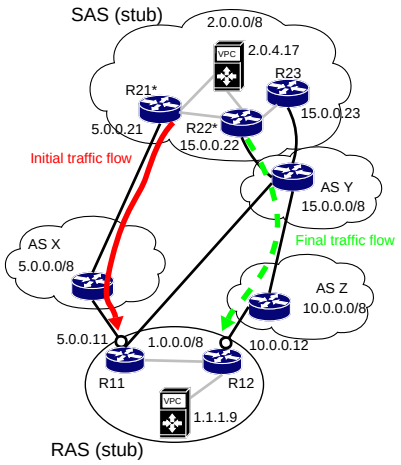

Fig. 2. Interdomain network topology

To initiate the establishment of Virtual Peerings and to exchange the parameters between the involved VPCs, a protocol is required. Instead of defining a new signalling protocol, we propose to rely on the already deployed BGP protocol as a mean to exchange Virtual Peerings requests. The reason for this choice is that such a protocol requires few modifications to BGP and that it can be deployed incrementally. Our extensions solve the following issues. First, the VPC in an RAS must learn the IP address of the VPC in the SAS. Second, the VPC in an RAS needs a secure mean of requesting from a VPC in a remote AS the establishment and removal of Virtual Peerings. Then, the VPC in the SAS must communicate with the border routers of its AS in order to setup the requested tunnels. Finally, routes must be distributed inside the SAS in order to advertise the tunnels.

A. Advertisement of VPC addresses

The Virtual Peering Controller Advertisement (VPCA) is used to advertise the IP addresses of the VPCs that serve the SAS. Indeed, to request the establishment of a Virtual Peering with the SAS, the RAS needs to know the IP address of the SAS's VPC. If a small number of ASes want to use Virtual Peerings, these addresses could be distributed in an offline manner, by e-mail or by other means. However, as the number of participants grows, an automatic solution will be required.

We propose to distribute the SAS's VPC IP address inside the BGP Update messages originated by the SAS. The VPC IP address is encoded in a transitive extended community value. The extended community attribute is an optional attribute supported by all BGP implementations. It is already used to encode various types of optional information [11]. We define the Virtual Peerings extended community that contains the IP address of the VPC that is responsible for the associated prefix and a set of bit flags indicating the types of tunnels that can be used to establish a Virtual Peering. For redundancy reasons, an AS could use several VPCs. In this case, it simply attaches several Virtual Peerings extended communities to the BGP routes that it originates.

In the example of Fig. 2, the BGP routes towards the prefix `1.0.0.0/8` advertised by the RAS will contain the IP address `1.1.1.9` in the Virtual Peering community. The BGP routes advertised by the SAS for prefix `2.0.0.0/8` will contain `2.0.4.17`, the IP address of the SAS's VPC.

B. Establishment and removal of Virtual Peerings

To request the establishment of Virtual Peerings, some messages must be exchanged between the RAS's VPC and the SAS's VPC. Instead of defining a new signalling protocol from scratch to establish the Virtual Peerings, we use a multi-hop eBGP session between the two VPCs. We do not describe the details of establishing a BGP session in this paper, but refer the reader to the BGP specifications [12]. This session is used by the RAS's VPC to send messages to the SAS's VPC. We call this session the Virtual Peering Session in the remainder of this paper. The messages exchanged over a Virtual Peering Session are not propagated to other BGP routers.

Two types of BGP messages are exchanged over a Virtual Peering Session: Virtual Peering Establishment and Virtual Peering Removal. A Virtual Peering Establishment (VPE) is a BGP Update message sent in order to request the establishment of a Virtual Peering or to change the parameters of an existing Virtual Peering. The VPEs sent by the RAS's VPC over the Virtual Peering session must contain both the destination prefixes (`1.0.0.0/8` in the case of Fig. 2) and the information required to establish the tunnels including the tunnel tail-end. This information can be encoded by using the tunnel SAFI proposed in [13]. This proposal defines a new type of address family that allows to attach tunnel information to the advertised prefixes. The encoding proposed in [13] allows to specify the parameters for different types of tunnels. For example, a tunnel route indicating a GRE tunnel can contain the required key and the session ID while a tunnel route indicating an L2TPv3 tunnel will contain the required cookie. Furthermore, several types of tunnels can be attached to each tunnel route. When advertising tunnel routes, a VPC may request distinct Virtual Peerings by advertising different prefixes with different associated tunnel tail-ends.

A Virtual Peering Removal (VPR) is a BGP Withdraw message sent in order to shutdown an existing Virtual Peering. The VPRs sent by the RAS's VPC only contain the prefix for which the Virtual Peering must be shutdown and concern the same address family. When a VPR is received by a VPC, it contacts the tunnel head-ends to shutdown the tunnels for the given prefix.

C. Distribution of Virtual Peering routes within the domain

The Virtual Peering Tunnel Route (VPTR) is a normal BGP Update message that the VPC sends to the border routers in order to distribute the tunnel routes received in a VPE. The VPTRs are sent over the genuine iBGP sessions that the VPC has established with the border routers. Each VPTR contains a *prefix*, a *tunnel tail-end* and the type of tunnel requested. The VPTR uses a different address family, so that both the normal

IPv4 addresses and the tunnel routes can be advertised over a single BGP session.

The selection of the best border routers to serve as the Virtual Peering head-end in the SAS could depend on how the SAS wants to optimise its interdomain traffic. In practice, this decision will be taken by the VPC. Two approaches are possible. In the first approach, the VPC itself can learn the eBGP routes known by each border router. Since it has an iBGP session with each border router, this is easy. The VPC can then measure the quality of each eBGP route based on predefined criteria by requesting each border router to perform latency and bandwidth measurement using a technique such as [14]. For instance, it can measure the latency of the routes or the maximum bandwidth available along the route. Based on the result of the measurement, the VPC can then select a single border router.

The second approach consists in establishing multiple tunnels. The VPC must then select multiple border routers that will serve as tunnel head-ends. This approach is interesting if the SAS has multiple peerings with its providers, located at very distant places. In this case, it can be interesting to setup tunnels departing at each of these peerings in order to favour hot-potato routing.

Upon reception of a VPTR each border router determines whether it has a best eBGP route to reach the *tunnel tail-end* in its BGP routing table. In that case, the border can serve as a tunnel head-end for the packets sent towards this *prefix*. For instance, in the example of Fig. 2, `R22` has learned an eBGP route towards the prefix of the tail-end router, `10.0.0.0/8`. Once the tunnel is established, the border router advertises via iBGP a new route indicating that it can reach the destination *prefix*. This route has a higher value of its `local-pref` attribute to force other iBGP neighbours to prefer it over routes received outside of the Virtual Peering. The AS-Path of this advertisement contains the AS-Path of the route that the border router uses to reach the tunnel tail-end. If there are no eBGP routes to reach the tunnel tail-end (this may be due to BGP policies), the border router will not serve as a tunnel head-end for the packets sent towards this *prefix*.

We assume that in a typical IP network, only a fraction of the border routers will be able to serve as tunnel head-ends. This could for example depend on the type of interfaces installed on each router. To allow the VPC to know the routers that are capable of establishing virtual peering links, we assume that each router indicates in the IGP link state packets that it originates the types of tunnels that it supports, if any. For IS-IS, this can be encoded by using the capability TLV defined in [15]. Based on its link state database, the VPC can thus easily determine the capabilities of all the border routers inside its AS.

D. Security considerations

From a security viewpoint, the Virtual Peerings approach proposed in this section exhibits two major issues. First, the VPCA message advertises to the global Internet the IP addresses of the VPCs attached to a prefix. If an attacker could modify the content of the Virtual Peerings extended communities in BGP advertisements passing through a (transit) router, it could redirect Virtual Peering requests to another machine. This could lead to traffic redirection attacks. However, it should be noted that if an attacker is able to modify BGP messages, many types of attacks are possible with the standard BGP that is deployed today. In order to avoid this problem, the best solution is to use one of the BGP extensions proposed in [16], [17], that allow to authenticate BGP advertisements. If those extensions cannot be used, a possible solution is to ensure that the IP address of the VPC belongs to the advertised prefix.

A second issue is due to the VPEs. When receiving a VPE, a VPC should be able to verify that the RAS is authorised to advertise those prefixes and tunnel tail-ends. Otherwise, an attacker could easily redirect packets sent by the SAS to its premisses instead of a tail-end in the RAS. This verification could be based on publicly available address allocation registries such as ARIN or RIPE. Several major ISPs, notably in Europe, already use those databases to filter the routes advertised by their peers and customers. Those techniques can also be used by VPCs. In the long term, the BGP security extensions being developed by the IETF [16], [17] will address this problem.

E. Deployment

Our rationale for designing the protocol described in this section was to make possible an incremental deployment. Since the protocol does not require modifications in the intermediate ASes, two domains can start to use it to negotiate Virtual Peerings. Moreover, inside a single domain, only a subset of the border routers must be updated in order to support the VPTRs. In addition, the VPC can initially be implemented in a separate workstation and later be deployed inside a genuine BGP router or route-reflector. Finally, while the BGP security extensions [16], [17] have not yet been deployed in the global Internet, it would be easier to use them between VPCs as there will be fewer VPCs than normal BGP routers and the VPCs do not redistribute the VPEs received over the Virtual Peerings sessions to other ASes.

A possible deployment scheme would be to initially start using Virtual Peerings between a small number of universities or research labs. Afterwards, the solution can naturally be deployed by content and access providers as well.

IV. Applications of Virtual Peerings

The Virtual Peerings can be used to achieve various types of traffic engineering objectives such as load balancing, preferring paths with the lowest delay or the highest bandwidth or reducing the cost of the traffic. We discuss in this section one of their utilizations: inbound traffic engineering.

Virtual Peerings can be used to gain a better control on the inbound traffic of a domain. As explained earlier, the traffic of a multi-homed stub domain is often imbalanced, a small number of ASes are responsible for a large fraction of the received traffic [4], [18], [6]. Virtual Peerings represent

a deterministic solution to the problem of inbound traffic engineering. When a stub AS is connected to two transit providers, it may, depending on the BGP configurations of its providers receive 80 or 90% of its traffic through one provider. This imbalance may lead to congestion and packet losses on the access links. To avoid this congestion, stub ASes need to move some incoming traffic flows between providers to obtain a better traffic balance.

In order to achieve a good balancing of the inbound traffic, the stub AS needs to monitor the traffic received on each access link. This can be done by activating NetFlow on the border routers' interfaces and by collecting the traffic statistics in a dedicated workstation [10]. Then, based on the traffic statistics combined with an optimisation algorithm, the stub AS identifies the source ASes that must be moved. For each source AS concerned, a Virtual Peering is established. Through the Virtual Peering, the destination requests the source AS to encapsulate its traffic in a tunnel towards a designated access link. We show later that this solution is feasible and that with a limited number of Virtual Peerings, it is possible to reach a near perfect load balance of the inbound traffic.

V. Virtual peerings for traffic balancing

To be useable as a technique to balance the incoming traffic of a multi-homed stub ISP, it should be possible for a stub to balance its incoming traffic by establishing a small number of Virtual Peerings although there are more than 16,000 ASes in today's Internet.

A. Simulation scenario

We use the Internet topology inferred by [19] from real BGP routing tables gathered from multiple vantage points, mostly in the Internet core. The topology dates from February 10th, 2004 and contains 16,921 domains and 37,271 interdomain links. There is at most one link between two different domains and each link represents the business relationship that exists between the two domains it connects. The possible relationships are *customer-provider* relationship where a customer buys connectivity from a provider and *peer-to-peer* relationship where the connection cost is shared by the two domains.

We use C-BGP [20] and model each domain as a single BGP router. We translate the business relationships between the domains into routing policies configured in each routers. These policies are composed of two parts. The first part enforces the so-called *selective export rule* [21] which governs the provision of transit. One domain will typically provide a full transit service to its customers, a limited transit service to its peers but never between its providers or its peers. The second part of the policies enforces the preference that a domain has for routes learned over different relations [21]. The routes learned from customers are preferred over routes learned from peers which in turn are preferred over routes learned from providers.

To ensure a good representativity, our simulations are performed for a large number of stubs. We consider 1000 dual-homed stubs, 1000 3-homed stubs, 295 4-homed stubs, 101 5-homed stubs and 49 6-homed stubs. This corresponds to 29% of the multi-homed stubs in today's Internet.

To model the traffic distribution in this topology, we assign traffic on all the sources following a Weibull distribution with shape parameter α equal to 0.5. With this distribution, about 1000 sources are responsible for 95% of the traffic received by a stub AS. This fits very well the traffic distribution shown in [6] and [22] and the references therein.

Based on those traffic distributions, we used C-BGP to compute the distribution of the incoming traffic on each of the considered 2445 stub domains. Fig. 3 shows the distributions of the traffic imbalance among all those stub domains. We define the traffic imbalance as the traffic volume received by the most loaded provider divided by the mean traffic volume. On the y-axis, we show the cumulative fraction of stubs that have the corresponding imbalance. Fig. 3 shows the traffic imbalance for stubs that have 2 to 6 providers. We observe on the first curve that less than 25% of dual-homed stubs have their inbound traffic well balanced over their providers, that is with an imbalance smaller than 1.01. Moreover, more than 35% of dual-homed stubs have an imbalance superior to 1.2, which means that 35% of the dual-homed stubs have one provider that receives more than 60% of the traffic. Among stubs that have 3 providers, about 60% have an imbalance larger than 1.2. Discussions with ISPs reveal that such large traffic imbalances are common.

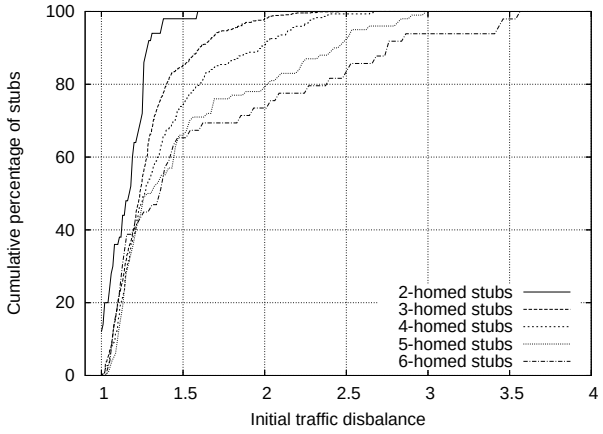

Fig. 3. Initial traffi c imbalance

B. Selection of the Virtual Peerings

Searching for a better repartition of the traffic is an optimisation problem which consists in allocating an access link to each source of traffic. This is a combinatorial problem. Several techniques can be used to solve this problem. We choose to use Evolutionary Computing techniques [23] implemented with the help of the GAUL library [24]. We choose an evolutionary algorithm to solve this problem because it is possible to extend it to support multiple objectives [25] and this technique has already been applied to solve different interdomain traffic engineering problems [22].

Our evolutionary algorithm (Alg. 1) relies on a population of individuals, that is, a set of potential solutions that evolves in time. In our population, an individual represents a particular assignment of the N sources on the M access links of the destination. An individual is thus an array $(l_0, \ldots, l_{N-1})$ of N integers $0 \leq l_j < M$ where each p_j is the identifier of the access link used by source j to enter the network. We initialize the population with individuals that represent the initial BGP situation, that is, an individual represents the access link used by the N sources. In practice, a stub network does not need to know the interdomain paths used by each source AS. It can use NetFlow on its border routers and collect the traffic statistics [10] to determine which source AS is received over which access link. Before starting the optimization, the algorithm slightly perturbates the initial individuals. In this way, we do not start with a population of identical individuals. The perturbation of an individual consists in replacing the access link of a randomly chosen source by a randomly chosen access link.

Alg. 1 Optimization algorithm

Let N be the number of sources
Let M be the number of access links
Let $(l_j)_{0 \leq j < N}$ be the initial access link used by each source
Let $(v_j)_{0 \leq j < N}$ be the traffic volume sent by each source
{Initialize population with $2N$ individuals}
$pop \leftarrow \emptyset$
for $k = 0$ to $2N - 1$ **do**
  $pop \leftarrow pop \cup mutate((l_0, ..., l_{N-1}))$
end for
{Evaluate fitness of individuals}
$evaluate\_fitness\_pop(pop)$
{Main loop, each generation updates the population}
while $(generation < MAXGEN)$ **do**
  {Crossover with probability 0.1}
  $crossover\_pop(pop)$
  {Mutation of individuals with probability 0.9}
  $mutate\_pop(pop)$
  {Evaluate fitness of individuals}
  $evaluate\_fitness\_pop(pop)$
  {Terminates if a good individual is found}
  if ($\exists k : pop[k]$ satisfies termination criterion) **then**
    break
  end if
  {Select best individuals based on fitness}
  $select(pop)$
end while

We fixed the population size, in an empirical manner, to twice the number of considered sources. The number of considered sources depends on the traffic volume distribution. In these results, the algorithm considers as many sources as required to cover 95% of the total traffic volume. That is, $N = 942$ sources were taken into account. During the evolution of the population, we perform mutations and crossovers. In our algorithm, a mutation consists in changing the access link of a randomly selected source (see Alg. 2). We refer the reader to [23] for an explanation of the crossover operation.

Alg. 2 Mutate individual $(l_0, ..., l_{N-1})$

{Choose a random access link i}
$i = rand(M)$
{Choose a random source j}
$j = rand(N)$
{The new access link used by source j is i}
$l_j \leftarrow i$

After each generation of the population, individuals are evaluated with a fitness function. The fitness function used in our algorithm (Alg. 3) measures for an individual the deviation that it causes in term of load balancing. Formally, in order to measure the fitness of an individual x, the algorithm first computes the percentage of traffic L_i that would be received by each access link i if the configuration represented by individual x was implemented with Virtual Peerings. Then, the function computes the L2 distance between the vector $(L_i)_{0 \leq i < M}$ and the equilibrium. The equilibrium represents the case where each access link receives an equal percentage of traffic $1/M$.

Finally, a selection is performed based on the fitness of individuals. The individuals that best fit the objective are kept while others are discarded.

Alg. 3 Compute the fitness of individual $(l_0, ..., l_{N-1})$

{Compute the load vector $(L_i)_{0 \leq i < M}$}
for $i = 0$ to $M - 1$ **do**
  $L_i \leftarrow \frac{\sum_{\forall j : l_j = i} v_j}{\sum_{0 \leq j < N} v_j}$
end for
{Compute the L2 distance from the equilibrium vector}
$fitness \leftarrow \sum_{0 \leq i < M} \left(L_i - \frac{1}{M}\right)^2$

C. Results

We used this evolutionary algorithm to determine the Virtual Peerings that each of our 2445 considered stubs would have to establish to approach a perfect balance among its access links by less than 1%. This means that in the case of a dual-homed stub for instance, the number of tunnels required causes the most loaded provider to carry at most 50.9% of the traffic volume. Figure 4 reports the cumulative distribution of the number of Virtual Peerings established by all those stub domains to approach of the perfect balance by less than 1%. We observe that in the case of dual-homed stubs, the objective is reached with no more than 41 tunnels for 90% of the stubs. In the case of 3-homed stubs, no more than 42 tunnels are required to balance the traffic of 90% of the stubs. Finally, less than 50 tunnels are required to balance the traffic among the providers of 90% of the 4-homed stubs.

We studied the sensitivity of the technique to the traffic distribution. We performed the same simulation with a traffic distribution that follows a Weibull with parameter $\alpha = 0.25$.

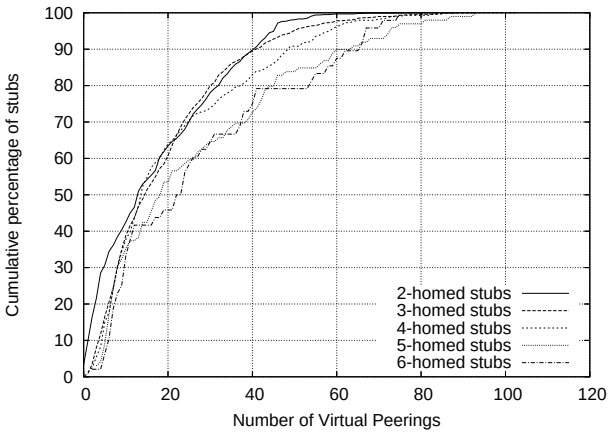

Fig. 4. Number of Virtual Peerings to establish

In this case, the situation is really unfavourable : 3000 source ASes produce 95% of the traffic received by a stub. The results of this simulation (not shown here due to space limitations) reveal that the number of Virtual Peerings required to balance the inbound traffic increases, but remains quite low. With $\alpha = 0.5$, a near perfect load balancing was possible with as few as 43 tunnels for 90% of the stubs. The remaining 10% of the stubs require between 44 and 94 tunnels. With $\alpha = 0.25$, 43 tunnels allow a near perfect load balancing of 68% of the stubs. Up to 80 tunnels are required to cover 90% of the stubs. The remaining 10% of the stubs still only need not more than 148 tunnels to balance their inbound traffic.

These figures show that an inbound traffic engineering relying on Virtual Peerings is feasible even with an unfavourable traffic distribution. Moreover, with Multi-Objective Evolutionary Computing [25] it would be possible to determine the optimal Virtual Peerings that minimise both the imbalance and the number of tunnels to establish. It would also be possible to combine load balancing with other objectives such as the latency reduction but we leave this as further work.

VI. Related work

Several works published in the literature have discussed interdomain traffic engineering. However, most of them focus on outgoing traffic engineering [3], [4], [5], [6], [18]. A few studies have discussed the performance of various "BGP-hacks" that are used by ASes to control their incoming traffic. In [26], we showed by simulations that AS-path prepending, although a widely used technique, provides a too coarse and non deterministic control on the incoming traffic. An alternative approach is to rely on techniques based on special BGP communities [7], [27]. Those techniques provide a finer control on the incoming traffic but are difficult to use in practice.

The idea of "negotiating" interdomain traffic engineering is also discussed in [28] and [29]. However, those solutions are limited to neighbouring ASes. Another approach is the utilisation of endsystem-based overlay networks such as RON [30]. In those approaches, overlays are established between endsystems based on collected measurements. To the contrary of overlays, our approach relies on tunnels that are established and maintained by the border routers of the source and destination ASes. Thus, our solution requires fewer tunnels and no modifications to the endsystems.

Our approach is close to the Detours proposed in [31]. Detours also relies on tunnels established between routers and assumes that endsystems are able to select the appropriate detour router. Our solution is completely transparent for the endsystems and we have shown how BGP can be used to establish the tunnels between routers.

Our approach has similarities to IPv6 multi-homing solutions (see [32] and references therein). With IPv6 multi-homing, each endsystem receives several IPv6 addresses, one per provider of its AS. By selecting the address that it uses to reach a destination, each host can indirectly select the interdomain path used. With IPv4, it is impossible to allocate several IP addresses to each host, but our Virtual Peering allows to control the flow of the incoming packets by terminating the tunnel at IP addresses belonging to the providers' CIDR blocks.

A systematic analysis of multi-homing is presented in [33]. This paper provides an in-depth analysis of the performance and reliability improvements that comes from multi-homing. In [34] a comparison of the relative benefits of overlay routing and route control in the case of multi-homing is made. In particular, it shows that route control with multi-homing can achieve performances that are close to overlay routing.

In [35], a load-balancing system is proposed an evaluated. It allows to control the traffic in both directions. As it relies on NAT, we do not consider this system to be applicable for large stub ASes such as broadband access providers. The measurement part of [35] could be combined with our approach. Several commercial multi-homing techniques have also been proposed recently, but few details are available about their operation [3], [36].

In addition, there are also proposals to bring drastic changes to interdomain routing. For instance, [37] considered the use of a separate protocol to carry control information and [29] proposes to introduce negotiation between ISPs. Unfortunately, to be used, those protocols and mechanisms must be supported by all transit ASes. This requires changes to potentially all BGP routers in the global Internet. Our approach relies on a cooperation between the source and the destination AS, but does not require any change to the transit ASes.

VII. Conclusion

Autonomous Systems that are packet consumers, such as ADSL or CATV providers, often need to control the flow of their incoming interdomain traffic for cost or performance reasons. In this paper, we have proposed the utilization of Virtual Peering to provide a deterministic and incrementally deployable solution to this problem.

A Virtual Peering is a unidirectional IP tunnel between a border router chosen by the source AS and a border router

chosen by the destination AS. Our solution to establish a Virtual Peering relies on three basic principles. First, there is a Virtual Peering Controller inside each AS and its IP address is attached as a BGP extended community to all BGP advertisements originated by the AS. Second, a multi-hop eBGP session is established between the VPCs of the source and destination ASes to negotiate Virtual Peerings. The source AS selects the head-end of the Virtual Peering based on its own traffic engineering objectives. Third, the destination AS selects autonomously the tail-end of the Virtual Peering. A key advantage of our approach is that it can be incrementally deployed inside cooperating stub ASes and does not require any change to the transit ASes. Given the size of the global Internet and the number of BGP routes, this incremental deployment is a key operational problem that must be considered.

As an example application of those Virtual Peerings, we have used a load-balancing problem. Our simulations have shown that in today's Internet, about 50% of the multi-homed stub ASes can balance their incoming traffic by requesting less than 20 Virtual Peerings. Furthermore, with only 70 Virtual Peerings, the incoming traffic of almost all stubs is well balanced. Our further work includes the definition of algorithms to select the optimal Virtual Peerings to meet other traffic engineering objectives such as reducing the end-to-end delay or the cost of interdomain traffic received by an AS.

ACKNOWLEDGEMENTS

This work was supported by the Walloon Government (DGTRE) within the TOTEM project (http://totem.info.ucl.ac.be). We are grateful to Steve Uhlig and Cristel Pelsser for their comments and suggestions.

REFERENCES

[1] J. Stewart, *BGP4 : interdomain routing in the Internet*, Addison Wesley, 1999.

[2] B. Halabi and D. Mc Pherson, *Internet Routing Architectures (2nd Edition)*, Cisco Press, January 2000.

[3] D. Allen, "NPN: Multihoming and route optimization: Finding the best way home," *Network Magazine*, February 2002, available from http://www.networkmagazine.com/article/NMG20020206S0004.

[4] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure, "Interdomain traffic engineering with BGP," *IEEE Communications Magazine*, May 2003.

[5] S. Uhlig and O. Bonaventure, "Designing BGP-based outbound traffic engineering techniques for stub ASes," *Comput. Commun. Rev.*, vol. 34, no. 5, 2004.

[6] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communications Review*, October 2003.

[7] B. Quoitin, S. Tandel, S. Uhlig, and O. Bonaventure, "Interdomain Traffic Engineering with Redistribution Communities," *Computer Communications Journal (Elsevier)*, vol. 27, pp. 355–363, March 2004.

[8] S. Bartholomew, "The art of peering," *BT Technology Journal*, vol. 18, no. 3, July 2000.

[9] Juniper Networks, "Ip services pics : datasheet," http://www.juniper.net/products/modules/100048.html, 2004.

[10] G. Varghese and C. Estan, "The Measurement Manifesto," *ACM SIGCOMM Computer Communications Review*, vol. 34, pp. 9–14, 2004.

[11] Srihari R. Sangli, Daniel Tappan, and Yakov Rekhter, "BGP Extended Communities Attribute," Internet draft, draft-ietf-idr-bgp-ext-communities-06, work in progress, August 2003.

[12] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," Internet draft, draft-ietf-idr-bgp4-26.txt, work in progress, October 2004.

[13] Gargi Nalawade, Ruchi Kapoor, and Dan Tappan, "Tunnel SAFI," Internet Draft, draft-nalawade-kapoor-tunnel-safi-01, work in progress, October 2003.

[14] CISCO Systems, "CISCO IOS Service Assurance Agent," http://www.cisco.com/warp/public/cc/pd/iosw/prodlit/saang_ds.pdf, 2004.

[15] J. Vasseur, S. Previdi, and M. Shand, "IS-IS extensions for advertising router capabilities," Internet draft, draft-vasseur-isis-caps-00.txt, work in progress, February 2004.

[16] Russ White, "Securing BGP Through Secure Origin BGP," *The Internet Protocol Journal*, vol. 6, pp. 15–22, June 2003.

[17] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, April 2000.

[18] S. Uhlig and O. Bonaventure, "Implications of Interdomain Traffic Characteristics on Traffic Engineering," in *European Transactions on Telecommunications, special issue on traffic engineering*, 2002.

[19] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points," in *INFOCOM 2002*, June 2002.

[20] B. Quoitin, "C-BGP, an efficient BGP simulator," `http://cbgp.info.ucl.ac.be`, September 2003.

[21] L. Gao, "On Inferring Autonomous System Relationships in the Internet," *IEEE Global Internet*, November 2000.

[22] S. Uhlig, O. Bonaventure, and B. Quoitin, "Interdomain Traffic Engineering with minimal BGP configurations," in *Proceedings of the 18th International Teletraffic Congress, ITC*, September 2003.

[23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.

[24] S. Adcock, "GAUL, the Genetic Algorithm Utility Library," `http://gaul.sourceforge.net`, 2004.

[25] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, June 2001.

[26] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin (Eds.), A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel, and S. Uhlig, "Internet Traffic Engineering," Chapter of COST263 final report, Springer-Verlag, October 2003.

[27] S. Agarwal and T. Griffin, "BGP Proxy Community Community," Internet draft, draft-agarwal-bgp-proxy-community-00, work in progress, January 2004.

[28] Jared Winick, Sugih Jamin, and Jennifer Rexford, "Traffic Engineering Between Neighboring Domains," Tech. Rep., June 2002.

[29] R. Mahajan, D. Wetherall, and T. Anderson, "Negociation-Based Routing Between Neighboring ISPs," in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation*, April 2005.

[30] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the eighteenth ACM symposium on Operating systems principles*, 2001, pp. 131–145.

[31] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: Informed internet routing and transport," *IEEE Micro*, vol. 19, no. 1, pp. 50–59, 1999.

[32] C. de Launois, O. Bonaventure, and M. Lobelle, "The NAROS Approach for IPv6 Multi-homing with Traffic Engineering," in *Proceedings of QoFIS, LNCS 2811, Springer-Verlag*, October 2003, pp. 112–121.

[33] A. Akella, B. Maggs, A. Seshan, A. Shaikh, and R. Sitaraman, "A Measurement-based Analysis of Multihoming," in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[34] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh, "A Comparison of Overlay Routing and Multihoming Route Control," in *Proceedings of ACM SIGCOMM*, Portland, OR, August 2004.

[35] F. Guo, J. Chen, W. Li, and T. Chiueh, "Experiences in Building a Multihoming Load Balancing System," in *Proceedings of IEEE INFOCOM*, March 2004.

[36] Cisco Systems, "Cisco Optimized Edge Routing," Tech. Rep., May 2004.

[37] S. Agarwal, C. Chuah, and R. Katz, "OPCA: Robust Interdomain Policy Routing and Traffic Control," in *Proceedings of the 6th International Conference on Open Architecture and Network Programming, IEEE OpenArch*, April 2003.