Morpheus: Making Routing Programmable

Yi Wang, Ioannis Avramopoulos, Jennifer Rexford
Princeton University
{yiwang, iavramop, jrex}@cs.princeton.edu

Abstract

Internet Service Providers (ISPs) express complex policies, affecting everything from business relationships with their neighbors to traffic engineering, scalability, and security, by configuring the Border Gateway Protocol (BGP). However, the routing architecture within an ISP, coupled with the multi-step BGP route-selection algorithm running on the routers, imposes significant restrictions on the policies that can be realized in practice. In this paper, we present Morpheus, a modular, open routing platform that addresses these limitations by changing the way BGP routes are propagated and selected within an ISP. With Morpheus, network operators can realize many useful policies that are infeasible today through flexible composition of multiple (possibly third-party developed) policy modules, and programming the route-selection algorithms. Morpheus can be readily deployed without requiring changes in other domains.

1 Introduction

Interdomain routing policies play a critical role in many aspects of Internet Service Provider (ISP) backbone management, including the business relationships with neighboring domains, the end-to-end performance offered to customers, the security of the network infrastructure and its customers, and the scalability of the routing protocols [6]. Collectively, these policy objectives determine which routes the ISP uses, and which neighboring domains are permitted to use these routes. For example, an ISP typically selects a route through a customer network, even if a shorter path to the destination exists through one of its peers or providers. As another example, if a peering link is congested, an ISP may adjust its policies to direct some traffic through a different peer. In this paper, we propose Morpheus, a modular, prorgammable routing platform that enables an ISP to realize useful policies that are infeasible in today's routing system, without requiring changes to other Autonomous Systems (ASes).

1.1 Limitations of the Routing Architecture

Today, an ISP expresses its policies by configuring the Border Gateway Protocol (BGP). The use of a single-path, path-vector routing protocol and hop-by-hop forwarding imposes several restrictions on how routes are selected *within a single AS* that fundamentally limit the policies an ISP can realize:

Propagating only one best route: Despite learning multiple routes for the same prefix, a BGP-speaking router only announces a single best route to its neighbors, making the rest of the candidate routes invisible to other routers. This restriction precludes each router from making its own independent choice from the set of candidate routes.

Selecting only one best route: Each router can only select one BGP route for forwarding data traffic. This not only limits the ability of routers to balance load over multiple paths, but also precludes an edge router from offering different routes to different customers.

Coupling of decisions across routers: Today, traffic entering the AS is forwarded to egress points in a hop-by-hop fashion. Edge routers connected to the same internal router are forced to direct traffic toward the same egress point.

We propose to overcome these limitations by ensuring *full visibility* into all candidate routes and *flexible assignment* of routes to routers. All BGP routes are propagated to a small collection of servers that make decisions on behalf of the routers; lightweight tunneling allows independent assignment of BGP routes to each ingress router (or link). As discussed in Section 2, these architectural changes allow Morpheus to support more flexible routing policies.

1.2 Limitations of the BGP Protocol

In addition to the intrinsic limitations of the routing architecture within an AS, the current BGP standard [13] and its *de facto* implementations of BGP also impose restrictions on the set of policies that can be realized, for three main reasons:

Overloading of BGP attributes: Today, many different policy objectives are intertwined into a few BGP attributes (e.g., "local preference", used to enforce business relationships and perform traffic engineering). Overloading of attributes makes it difficult to incorporate new policy objectives without modifying the configuration of existing ones.

Difficulty in incorporating "side information": Policy objectives often depend on external information, like measurement data or business relationships with neighbor ASes. Satisfying policy objectives also sometimes requires updating state, such as a history of (prefix, origin AS) pairs or statistics about route instability, over time. However, importing and updating state is very difficult today.

Restrictive step-by-step route-selection algorithm: The BGP route-selection algorithm selects the best route from all candidate routes by considering one attribute at a time (e.g., first local-preference, then AS-path length, and so on). This strict prioritization of BGP attributes limits ISPs to policies

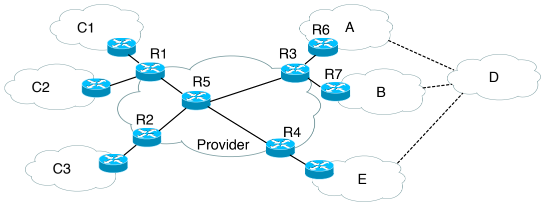

Figure 1: Example illustrating architectural principles

that rank one attribute over another, precluding policies that try to strike a balance between different policy objectives.

In this paper, we address these limitations by supporting *composition* of independent (possibly third-party developed) modules for different policy objectives, through a *programmable* route-selection process running on the servers. This affects the software architecture of the servers that make the routing decisions, as discussed in Section 3.

We call our server software Morpheus, since it is a modular, open routing platform that gives network operators the power to "shape" their routing policies. In Section 4, we present the major design decisions in our initial prototype implementation. To illustrate the expressiveness of our system, Section 5 describes several example policies that Morpheus can achieve that are not possible today. We discuss related work in Section 6 and conclude the paper in Section 7.

2 Routing Architecture for Programmability

In this section, we discuss three principles of routing architecture within an AS that are essential in enabling flexible interdomain routing. We also describe how Morpheus realizes each of these principles, providing important background material for the rest of the paper.

2.1 Complete Visibility of BGP Routes

An edge router may learn multiple routes for the same destination prefix, through BGP sessions with neighbor ASes and internal routers. However, BGP requires the router to select and propagate at most one route (for each prefix). As a result, this router may learn many routes that are never seen by any other routers in the AS. For example, in Figure 1 router R3 learns two routes to destination D but only propagates the one (say, via R6) to R1. Then, R1 cannot learn or use the other route (via R7), even if R1 might have preferred this route (say, because it is shorter or circumvents some particular AS) or wanted to split traffic over both routes. Although propagating only one route helps limit control-plane overhead, it imposes severe constraints on routing policies.

Principle: *An AS should have complete visibility of eBGP-learned routes to enable flexible routing policies.*

Morhpeus achieves full visibility by directing all external BGP (eBGP) routes to a small collection of servers that make decisions on behalf of the routers, as shown in Figure 2 and inspired by earlier work on the Routing Control Platform (RCP) [7]. Each server has a (multi-hop) eBGP session with each external neighbor router, in lieu of having direct eBGP sessions between the edge routers in the two ASes. Morpheus assigns a BGP route for each prefix to every internal router individually, using internal BGP (iBGP) sessions for backwards compatibility. Since the routers are no longer responsible for propagating BGP routing information to neighbor ASes, Morpheus does not need to send all of the route attributes—only the destination prefix and next-hop address are strictly necessary. This enables a significant reduction in the amount of BGP information the routers must receive and store. Morpheus also ensures that the BGP routes propagated to eBGP neighbors are consistent with the route assigned to the associated edge routers, and include all route attributes expected by the neighbor ASes.

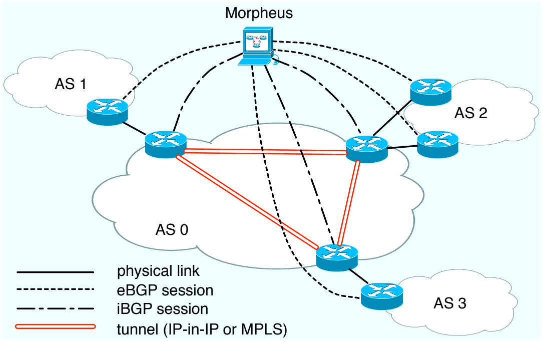

Figure 2: Morpheus server has BGP sessions with routers

2.2 Flexible Egress Selection Per Router

Although routers within an AS select best BGP routes in a distributed fashion, they cannot make *independent* decisions. This is because, under hop-by-hop forwarding, routers have to make consistent decisions to ensure consistent forwarding (i.e., avoid forwarding loops). To avoid inconsistent decisions, when a router has multiple "equally good" routes, it is common practice is to pick the route through the "closest" egress point, based on the Interior Gateway Protocol (IGP) weights, a.k.a. hot-potato routing. However, hot-potato routing introduces problems of its own. First, it significantly restricts the policies an AS can realize. For example, in Figure 1, R1 and R2 connect to a common intermediate router (R5). Hot-potato routing forces them to use the same egress point, rather than allowing (say) R1 to use R3 and R2 to use R4. In addition, a small IGP change can trigger routers to change egress points for many prefixes at once, leading to large traffic shifts and heavy processing demands on the routers [14].

Principle: *An edge router in an AS should be able to use any available path (i.e., egress point) independently.*

To achieve this goal, Morpheus relies on IP-in-IP tunnels or MPLS label-switched paths to direct traffic between edge routers, as shown in Figure 2. This design choice offers several important advantages. First, Morpheus can freely assign different BGP routes to different edge routers, without concern for inconsistent forwarding. Second, Morpheus can rely on the IGP to determine how traffic flows between ingress and egress routers, reducing the complexity of Morpheus and ensuring fast reaction to internal topology changes. Third, Morpheus does not select BGP routes for the internal routers, reducing the total number of routers it has to manage. Fourth, tunneling also allows the ISP to configure the ranking of egress points for each ingress router to achieve traffic-engineering goals. These *stateless* tunneling technology is readily available at line rate in commercial routers supporting MPLS or IP-in-IP encapsulation, and a "BGP-free core" is increasingly common in large ISPs.

2.3 Multipath Routing and Forwarding

Today, an edge router exports the same best route to each of its eBGP neighbors, making many useful policies impossible. For example, a router cannot split traffic over multiple paths for more flexible load balancing, or allow different customers to use different paths. In Figure 1, two customers C1 and C2 connected to the same edge router R1 may want different paths to reach the same destination prefix D (e.g., C1 prefers the path through egress point R3, while C2 prefers the path through egress point R4). However, R1 is unable to satisfy both requirements today. It is worth pointing out that the ultimate flexibility of multipath routing and forwarding is the ability to select and use any available *egress link*, rather than *egress router*, because a single edge router may learn more than one route for the same prefix through different egress links, such as in Figure 1, where R3 can reach D through R6 or R7. The ISP should be able to assign these routes to different customers and use them *independently*. For instance, C1 should be able to use the path through link R3–R6, while C3 uses the path through link R3–R7.

Principle: *An edge router in an AS should be able to use multiple paths (i.e., egress links) to reach a destination.*

Support for multipath routing and forwarding would allow ISPs and their customers to capitalize on these diverse paths for better performance and reliability, and even better security [19]. It also ensures that each router has a backup path for faster failover (e.g., in case the next-hop address of the primary route becomes unreachable). With full visibility into the eBGP-learned routes and the internal topology, Morpheus can easily pick the best routes on behalf of every edge router and neighbor AS individually. Morpheus sends multiple routes to each edge router[1], and the "virtual routing and forwarding (VRF)" or "virtual router" features commonly available in commercial routers can be configured to allow different customers to use different paths [12]. To avoid requiring changes in neighbor ASes, we focus on a restricted form of multipath in which each eBGP neighbor of the ISP still learns just one route per prefix, while different eBGP neighbors may learn different routes.

3 Software Architecture for Programmability

In this section, we discuss three principles that drive the software architecture of Morpheus. Morpheus cleanly separates two operations that are intertwined today—classifying routes according to policy objectives and deciding how to weigh these objectives in picking the best routes. Policy classifiers are programmable modules that tag the routes, perhaps consulting or updating local state. Programmable route-selection algorithms allow network operators to make tradeoffs between policy objectives in selecting the best routes.

3.1 Separation of Policy Objectives

Routing policies must balance multiple objectives such as business relationships, AS path length, and traffic engineering. Today, policy objectives are usually translated into specific values of BGP attributes to influence the route-selection algorithm. For example, different business relationships (customer, peer, provider) are typically represented using different local-preference (`local-pref`) values. Unfortunately, today's BGP implementations do not provide an easy way to add new policy objectives into the system. As a result, multiple policy objectives must share a limited number of BGP attributes. For example, it is a common practice to also use `local-pref` to achieve traffic-engineering objectives. Overloading the attributes leads to complex, convoluted routing policies, where the network operators cannot easily reason about the policy objectives independently. The problem is exacerbated when operators need to add new policy objectives, further overloading the same attributes.

Principle: *Policy objectives should be expressed independently and have their own route attributes.*

Morpheus achieves this goal by implementing policy objectives as independent *import policy modules*, as shown in Figure 3. Each module works as a classifier for a particular policy objective. The module receives as input a route and produces as output a tag that is affixed to the route. For example, a business-relationship module may tag a route as "customer", "peer", or "provider"; a stability module (evaluating the stability of routes based, for example, on a route-flap damping algorithm) may tag a route with an number (e.g., an integer between 0 and 99), where a bigger number implies higher stability. Each import policy module has its own tag space, obviating the need to overload the same attributes. By tagging the routes, rather than filtering or suppressing them, the route-selection algorithm is guaranteed to have full visibility of the candidate routes. These tags are purely local to Morpheus, and are not disseminated to the routers; as such, using these tags does not require any changes to the BGP protocol implemented on the routers.

3.2 Incorporating "Side Information"

Many useful policy objectives require certain *side information*, including *external information* such as business relationships, measurement data, and registry of prefix ownership, and *internal states* such as a history of (prefix, origin AS) pairs. However, there is no systematic mechanism to

[1] This can be achieved by using the "route target" attributes commonly used with VRF in MPLS-VPN [12], or having multiple iBGP sessions between a Morpheus server and an edge router. Other options include using the BGP "add-paths" capability [18] or a new message dissemination protocol, which may be more efficient at the expense of backwards compatibility.

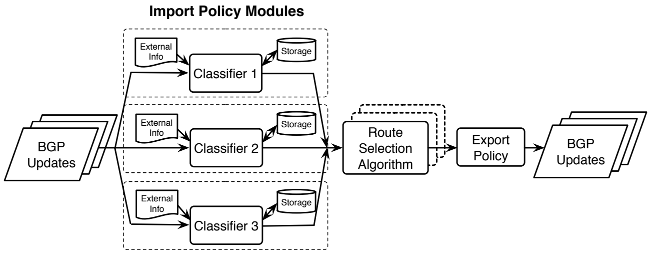

Figure 3: The modular architecture of Morpheus

incorporate side information to routers today. Network operators have to either "hack" their BGP configurations in an indirect and clumsy way (e.g., re-configuring filters and community attributes), or wait for software updates from router vendors (if the need for certain side information becomes compelling) and then upgrade a large number of routers.

Principle: *Flexible policies should be able to consult and update "side information" in classifying routes.*

In Morpheus, each policy classifier can import external information and update internal state. For example, the business relationships module can have access to up-to-date information about the ISP's business relationships with neighboring ASes through a configuration file, or a database. A security module can have access to a registry of prefixes and their corresponding owners. A performance module can get periodic updates from a monitoring system. A route stability module can maintain statistics about route announcement/withdrawl frequencies. A route security module that implements Pretty Good BGP—a simple algorithm that can effectively detect BGP prefix and subprefix hijacks—can keep past history of BGP updates in the past 24 hours [10].

3.3 Programmable Route Selection

Today's BGP route-selection algorithm applies a series of tie-breaking steps, one route attribute at a time. The ordering of the steps is built in to the routers and is relatively difficult to change, though some vendors enable operators to disable some steps. Imposing a strict priority on the attributes is especially restrictive, as it precludes policies that make trade-offs across different policy objectives.

Principle: *Flexible policies require a route-selection algorithm that can balance multiple policy objectives.*

Morpheus allows network operators to write their own route-selection algorithm that selects best routes based on the tags set by the import policy modules. However, many network operators would prefer not to write a "program" (e.g., in C or another higher-level language) every time they want to changes their routing policies. In addition, expressing arbitrary route-selection algorithms in a higher-level language introduces a variety of risks (e.g., that the program never terminates). Instead, we envision that operators would use a simple configuration interface to control how Morpheus weighs the policy objectives. The underlying (configurable) algorithm should be flexible enough to support most useful policies, and efficient enough to terminate quickly.

Furthermore, in order to realize such policies that require multipath routing, Morpheus supports the parallel execution of multiple route-selection algorithms. Each algorithm can be configured to realize a different policy and select a potentially different best route for the same prefix[2]. With this feature, an ISP running Morpheus can offer different types of routes to its customers as a revenue-generating service.

4 Prototype Implementation

Our Morpheus prototype is implemented as an extension to the XORP open-source software router [9], since it has a modular architecture well-suited to our design. However, since XORP is designed to operate as a single router, we had to modify it to pick (possibly different) BGP routes on behalf of many routers. Our design goal for the prototype is to demonstrate its flexibility in realizing policies that are infeasible today, while keeping the system simple and efficient.

4.1 Import Modules With Numerical Tags

To simplify the decision process, each import module assigns an integer tag (0 to 99) to each route, where a higher number implies greater preference. Restricting the tags to integer values does not compromise the flexibility of the import modules, since most policy objectives have simple rules for classifying the routes. For example,

Business relationships: The business-relationship module assigns different tags (say, 90, 40, and 10) to routes learned from customers, peers, and providers, respectively, by consulting a table of (AS number, business relationship) pairs.

Route stability: The route-stability module assigns a higher number to routes that are more stable, where the tag represents the inverse of the penalty function used in today's route-flap damping algorithm.

Suspicious routes: The Pretty Good BGP (PGBGP) [10] module[3] identifies suspicious routes based on a history of (prefix, origin AS) pairs. The module assigns a tag of 99 if a route's origin AS matches the history of the past 24 hours, or a 0 otherwise.

The import modules are connected in a pipeline, consistent with the XORP software architecture. The first module is a filtering module that discards BGP routes that should not be used under any circumstances (e.g., due to a loop in the AS path).

4.2 Score-Based Route-Selection Algorithms

Upon receiving a new route, a route-selection algorithm computes a *final score* S, as a function of the module-specific tags (s_i for policy objective i). The score function is given by the following formula:

$$S = \sum_{i=1}^{n} f(s_i) = \sum_{i=1}^{n} (a_i s_i + b_i) u[(s_i - c_i) d_i] \qquad (1)$$

This formula enables network operators to trade off different policy objectives by adjusting parameters a_i, b_i, c_i, and d_i.

[2] Alternatively, a route-selection algorithm can return a set of best routes (e.g., top *k* routes) instead of one, if desired.

[3] The PGBGP module was implemented by Josh Karlin.

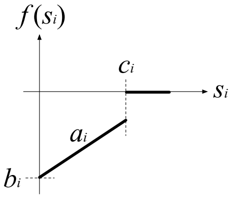

Figure 4: The $f(s_i)$ of the score function (with $d_i = -1$)

The effect of these parameters on the final score is given in Figure 4. Patameter a_i is called a *weight* and controls the slope of the score function. Parameter b_i is called an *offset* and guarantees that routes with s_i below the minimum or above the the maximum desirable value get lower final scores than routes in the opposite range. Parameter c_i is called a *threshold* and is used in expressing policies that need not distinguish between tag values above or below a given threshold. (For example, if the policy objective is stability, any route having a tag above, say, 70 might be viewed as "stable enough.") In order to support this functionality, the score function also contains the unit step function $u(x)$.[4] Parameter d_i is called a *sign* and is used to control the minimum or maximum desirable value for each tag.

This simple formula provides powerful support for trade-offs between policy objectives, while ensuring fast computation of a final score for each route. In addition, selecting the best route(s) requires only simple comparison operations to identify the route(s) with the maximum score. When a new route arrives, the route-selection algorithm only needs to compare the new route's final score to the current maximum value, instead of comparing to all candidate routes. The downside of our approach is that the network operator must tune a potentially large number of parameters and reason about their effects on route selection. (In our ongoing work, we are exploring existing techniques from decision theory, such as Multi-Criteria Decision Analysis [2, 15], to provide a framework for balancing trade-offs between multiple policy objectives.)

After comparing routes based on their final scores, a route-selection algorithm may find that multiple routes have the same maximum score. To compute a single best route for each edge router, Morpheus applies a rank-based tie-breaking algorithm. For each edge router, Morpheus has a fixed (but configurable) ranking of all egress points. This ranking may reflect geographic distance or the typical IGP distances (e.g., as reflected in the network design) between each pair of edge routers. By decoupling changes in the IGP distances from the BGP routing decisions, the fixed-ranking scheme avoids the problems associated with hot-potato routing [14] and gives the ISP additional control over the flow of traffic. A closer coupling with the IGP distances, where needed, can be achieved on a longer time scale by simply adjusting the configuration of the fixed ranking.

In the end, a route-selection algorithm selects a single best route for each destination prefix for each edge router and each eBGP neighbor. A limited, yet powerful, form of multipath routing is achieved by running multiple route-selection algorithms, each configured to emphasize different objectives. In particular, each eBGP neighbor can subscribe to the output of one route-selection algorithm, to learn and use the best routes that algorithm produces. We plan to extend our prototype to provide eBGP neighbors with finer-grain control over which prefixes are learned from each route-selection algorithm. This kind of multipath service can be readily deployed today without requiring changes to the routers used in other ASes.

5 Applications

With Morpheus, network operators can realize many useful policies that are *infeasible* today. In this section, we present two examples of such policies.

5.1 Trade-offs Amongst Multiple Criteria

In the first example, we give a configuration of our prototype that realizes the following policy (also mentioned previously in Section 1) that trades revenue for route stability:

"If all routes are somewhat unstable, pick the most stable path (of any length through any kind of neighbor), otherwise pick the shortest stable path through a customer."

Assume for simplicity that there are three import modules (policy objectives) in the system: one for business relationships, one for AS-path length, and one for route stability. The business-relationships and route-stability modules assign tags s_1 and s_2 to a route according to Section 4.1, respectively. The AS-path-length module assigns tag $s_3 = 100 - 4n$ to a route, where n is the AS-path length. Supposing a route is stable if and only if $s_2 \geq 70$, the above-mentioned policy can be realized by the following function:

$$S = 100s_1 + (10100s_2 - 1010000)u(70 - s_2) + s_3 \quad (2)$$

Variables $a_1 = 100$ and $a_3 = 1$ are chosen such that, for every s_1 and s_3, $a_1 s_1 > a_3 s_3$. This condition implies that, given the same stability tag, a customer route is guaranteed to get a higher final score than a peer or provider route irrespective of the length. Variable $a_2 = 10100$ is chosen in the same way. Finally, variable $b_2 = 1010000$ is chosen to guarantee that each stable route ($s_2 \geq 70$) has a higher final score than every unstable one ($s_2 < 70$).

5.2 Backward Compatible Multipath Service

In this example, Morpheus offers two types of paths: (a) "default path", for regular customers, and (b) "secure path", for customers who are willing to pay more to have the most secure paths available. Here by "secure", we mean a route with tag "99" assigned by the Pretty Good BGP (PGBGP) module. Customers can subscribe to the type of routes they want through some out-band channel (e.g., a Web page) and do not need to change their network settings.

Morpheus offers this service by running two separate route selection algorithms in parallel. One algorithm selects

[4] $u(x)$ is 0 if $x < 0$ and 1 otherwise.

the default paths by having a score function that generally balances different policy objectives (e.g., the one in Section 5.1). The other algorithm selects secure paths by setting the weight a_p of the PGBGP module tag s_p as:

$$a_p > 100 \sum_{i=1, i \neq p}^{n} a_i \tag{3}$$

This condition guarantees a route with the largest PGBGP tag s_p will have the highest final score, given s_i is integer between 0 and 99. Other policy objectives only take effect when there are more than one route with the same highest PGBGP score. Each algorithm selects its own set of best routes independently, and sends them to the export module. According to the customers' subscription, the export module sends the right type of routes to each customer AS. (Peer and provider ASes always receive default routes.)

6 Related Work

Previous work tried to raise the level of abstraction for configuring BGP policies through network-wide, vendor-neutral specification languages [1, 3]. However, we believe new languages are not sufficient, because today's intra-AS architecture introduces peculiar constraints on the combination of policies that can be realized. Several recent studies advocated moving the BGP control plane to a small set of servers [8, 4, 11, 7, 5, 16, 17]. These studies show that a logically-centralized control plane can prevent routing anomalies (such as protocol oscillation and forwarding loops) and provide fine-grain control over routing. They also showed that servers built from commodity hardware can be fast and reliable enough to control BGP routing in a large ISP backbone. We add to this body of work by showing how to support the flexible composition of multiple policy modules through a programmable route-selection algorithm. We also illustrate the importance of visibility, tunneling, and multipath routing in enabling flexible policies. Finally, our work relates to past work on extensible router software, such as XORP [9], though we focus on programmable routing policies with *AS-wide* route visibility and control.

7 Conclusions

Morpheus is a modular, open routing platform that enables network operators to compose multiple, independent policy objectives, without requiring the cooperation of other ASes or changes to the underlying routers. In our ongoing work, we are investigating Multi-Criteria Decision Analysis (MCDA) as a framework for balancing trade-offs between multiple policy objectives in route-selection algorithms. We also exploring ways to ensure the safety and efficiency of third-party modules for policy classification and route-selection algorithms, perhaps by imposing a restricted programming model.

Although most policy objectives can be implemented as import policy modules, traffic engineering is an exception. Traffic engineering typically depends on routing decisions across many routers and destination prefixes. We are exploring ways to embed the load-balancing objectives directly in the route-selection algorithm. In addition, we believe that multipath routing would make traffic engineering significantly easier, by allowing the routers to simply adjust the percentage of traffic they place on each path, rather than selecting an entirely new path for one or more prefixes.

Finally, we are studying the influence of flexible routing policies and multipath routing on global BGP convergence. We believe that many useful local policies can be achieved without compromising global stability, including policies that are not possible without deploying a system like Morpheus.

References

[1] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing policy specification language (RPSL). RFC 2622, June 1999.

[2] V. Belton. *Multiple Criteria Decision Analysis*. Springer, 2003.

[3] H. Boehm, A. Feldmann, O. Maennel, C. Reiser, and R. Volk. Network-wide inter-domain routing policies: Design and realization. *Draft*, April 2005.

[4] O. Bonaventure, S. Uhlig, and B. Quoitin. The case for more versatile BGP route reflectors. Expired Internet Draft draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.

[5] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a Routing Control Platform. In *Proc. Networked Systems Design and Implementation*, May 2005.

[6] M. Caesar and J. Rexford. BGP policies in ISP networks. *IEEE Network Magazine*, October 2005.

[7] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The case for separating routing from routers. In *Proc. ACM SIGCOMM Workshop on Future Direction in Network Architecture*, August 2004.

[8] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *Computer Communications Review*, 35(5):41–54, October 2005.

[9] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov. Designing extensible IP router software. In *Proc. Networked Systems Design and Implementation*, May 2005.

[10] J. Karlin, S. Forrest, and J. Rexford. Pretty good BGP: Improving BGP by cautiously adopting routes. In *Proc. International Conference on Network Protocols*, November 2006.

[11] T. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnai, and T. Woo. The SoftRouter architecture. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, November 2004.

[12] I. Pepelnjak and J. Guichard. *MPLS and VPN Architectures*. Cisco Press, 2000.

[13] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC 4271, January 2006.

[14] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, June 2004.

[15] E. Triantaphyllou. *Multi-Criteria Decision Making Methods: A comparative Study*. Springer, 2004.

[16] J. van der Merwe et al. Dynamic connectivity management with an intelligent route service control point. In *Proc. ACM SIGCOMM Workshop on Internet Network Management*, September 2006.

[17] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. van der Merwe. Wresting control from BGP: Scalable fine-grained route control. *Proc. USENIX*, 2007.

[18] D. Walton, A. Retana, and E. Chen. Advertisement of multiple paths in BGP. Internet Draft draft-walton-bgp-add-paths-05, March 2006.

[19] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford. Don't secure routing protocols, secure data delivery. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, November 2006.