Morpheus: Enabling Flexible Interdomain Routing Policies

Yi Wang Ioannis Avramopoulos Jennifer Rexford
Princeton University

Abstract

Giving ISPs more fine-grain control over interdomain routing policies would help them better manage their networks and offer value-added services to their customers. Unfortunately, the current BGP route-selection process imposes inherent restrictions on the policies an ISP can express, making many useful policies infeasible. In this paper, we present Morpheus, a routing control platform that enables a single ISP to realize a much broader range of routing policies without changing the underlying routers or coordinating with other domains. Inspired by multi-criteria decision analysis (MCDA), the design of the Morpheus server allows network operators to easily define new policy objectives, make flexible trade-offs between the objectives, and realize customer-specific policies. We present the design, implementation, and evaluation of Morpheus as an extension to the XORP software router. Our experiments show that Morpheus can support a large number of policies while handling the high rate of BGP update messages seen in large ISPs.

1 Introduction

BGP, the interdomain routing protocol for the Internet, was the first routing protocol to go beyond shortest-path routing to support flexible routing policies. Driven by the commercialization of the Internet, BGP was designed to allow each AS to select routes based on local policies. Today, BGP routing policy is used to achieve many different network management goals, such as implementing business relationships with neighboring domains, providing high quality end-to-end performance to customers, improving the scalability of the routing protocols, and protecting the network from attacks [5]. However, despite the goal of supporting flexible local policies, the BGP decision process and the configuration interface provided by router vendors impose unnatural restrictions on the way ISPs can select routes, for example:

- The BGP decision process running on each router selects a single "best" route for each prefix, forcing all neighboring ASes connected to the same edge router to learn the same route, even if some customers would be willing to pay more to use other routes.
- The BGP decision process imposes a strict ranking of the route attributes, where local preference has strict priority over AS-path length and so on. As a result, policies that strike a trade-off between different policy objectives are imposible to realize. For example, an AS cannot realize the following simple policy: *"If all routes are somewhat unstable, pick the most stable route (of any length through any kind of neighbor); otherwise pick the shortest stable route through a customer (then peer, and finally provider)."*

In this paper, we investigate how to support flexible routing policies while retaining the local autonomy ISPs have over route selection. We take today's interdomain routing system as a starting point, rather than designing a new architecture from scratch. In this setting, we focus on how a single ISP can realize more flexible policies. Although our design does not preclude, and, in fact, encourages cooperation among different ISPs, we believe that such cooperation should not be a prerequisite for supporting more flexible routing policies. Since ISPs are often business competitors, cooperation among them has proved notoriously difficult in practice. Fortunately, large ISPs have a lot of path diversity, and can safely and effectively "act alone" in applying more flexible routing policies, as discussed in Section 2. These observations lead us to design *Morpheus*, a routing control platform that gives individual ISPs the power to "shape" their routing policies.

The high-level problem Morpheus solves is how to assign a single interdomain route for each prefix to each edge router (or edge link), among the set of alternative routes learned from neighboring ASes. To maximize flexibility, Morpheus has complete visibility of the available routes and complete control in assigning routes to the routers, while remaining backwards compatible with existing routers, as discussed in Section 3. Our design of the Morpheus server is inspired by Multi-Criteria Decision Analysis (MCDA), a fast-growing area in decision theory that provides a formal basis for balancing trade-offs between different criteria. Thinking about interdomain route selection as an MCDA problem leads us to separate *route classification* (i.e., tagging routes based on policy objectives) from *route selection* (i.e., using the tags to compute a score, and selecting the highest-scoring route), as discussed in Section 4.

Our evaluation of Morpheus consists of two main parts. First, Section 5 shows that Morpheus can support a wide range of useful routing policies that are infeasible with today's monolithic BGP decision process. In fact, we show that Morpheus makes it possible to deploy several recently proposed extensions to BGP [10, 17, 15] without changing the legacy routers. Second, we present the implementation (Section 6) and evaluation (Section 7) of Morpheus as an extension to the XORP software router [12], to demonstrate that our system can handle the high rate of BGP update messages seen in large ISP networks, while supporting a large number of different policies in parallel. We discuss related work in Section 8 and conclude in Section 9.

2 Case for More Flexible Routing

In this section, we argue that more flexible control over interdomain routing policies would offer substantial benefits to large ISPs and their customers. We first argue that large ISPs typically learn several routes for each destination prefix and that these routes may differ substantially in security and performance properties. We then argue that the existing BGP decision process places unnecessary restrictions on the routing policies an ISP can realize. Finally, we show that an ISP can safely exploit extra flexibility without coordinating with other ASes.

Large ISPs have many routes to choose from.
Large ISPs that offer transit service usually connect to many neighboring ASes, often in multiple locations [19, 17]. For example, AS A in Figure 1 has four different router-level paths to D, through three different neighboring ASes. Various studies have quantified the rich path diversity seen by large ISPs. For example, at least 2% of all the ASes (which are likely to be tier-1 or tier-2 ASes) have ten or more unique AS paths for certain destinations [19]. A survey conducted in April 2007 on the NANOG mailing list shows that 5-10 router-level paths per prefix is quite common in large networks, with some prefixes having more than 20 different paths [20]. A detailed study of an individual ISP reported an average of 20 router-level paths for each prefix [34]. These statistics all suggest that large ISPs often have many downstream routes to choose from.

The paths can have different security and performance properties.
The many alternate routes a large ISP has can have different security and performance properties. In both cases, rich path diversity brings benefits.

Security: Prefix and sub-prefix hijacking, in which a prefix/sub-prefix is announced by an AS that does not legitimately own it (either maliciously or accidentally) can cause serious, even disastrous damage (e.g., in case of online banking) to network users [13]. It is recently shown that path diversity from a richly connected provider (e.g., tier-1) alone can be very effective in helping its customers resist prefix/sub-prefix hijacks, as it is very hard to hijack all the routes of a large ISP [37, 13].

Performance: Path performance (e.g., latency, loss, etc.) is another important factor ISPs should take into account when selecting routes, especially those ISPs that host real-time applications, such as voice-over-IP, video conferencing, or online gaming. However, the current BGP decision process considers little about path performance: the only relevant metric in the decision process—AS-path length—is a poor indicator of path performance [26, 28, 29]. As a result, alternate BGP paths often have significantly better performance that the default paths [8]. We believe large ISPs can select better performing paths by leveraging their path diversity [8]. Although some products exist in an multi-homed enterprise environment that offer performance enhancement by exploiting path diversity [7], there is no similar counterpart solution in large carrier ISPs.

Diverse path properties call for a more flexible decision process.
Although we use security and performance as examples in illustrating the benefits of rich path diversity, real world routing policies are far more complex, consisting of many different, sometimes conflicting *policy objectives*, such as business relationships, performance, security, stability, and traffic engineering. Given a set of available routes a large ISP has, it is possible that one route has the best performance, another route is most secure, yet another is most stable, i.e., there is no single "best" route that is perfect in every respect. Therefore, the ISP must synthesize its preferences ("weights") on each objective into an overall policy to select the best route.

However, the current BGP decision process imposes inherent restrictions on the policies an ISP can realize [23]. Consisting of a series of tie-breaking steps, the BGP decision process compares one attribute at a time until only one best route remains. The ordering of steps imposes a strict *ranking* on the route attributes, making it impossible to realize flexible policies that make *tradeoffs* between policy objectives. For example, a useful policy that strikes a balance between revenue and route stability could be: *"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 (then peer, and finally provider)."* However, this seemingly simple policy cannot even be expressed in today's router configurations. In addition, policy objectives that are not part of the original BGP

protocol, such as security and performance, are hard to add into its decision process, even if the importance of these objectives becomes obvious over time.

Different customers may want different routes.

Customers of a large ISP may have very different requirements on the types of routes they want. For example, customers in the financial industry may prefer the most secure routes, while customers that provide interactive applications may prioritize paths with low latency. If such options were available, they might be willing to pay more to have the routes they want. Yet there are many other customers who may be perfectly happy with any paths the ISP provides at a relatively low price.

Unfortunately, although large ISPs have the path diversity and strong economic incentive to provide customer-specific routes, they do not have the means to do it today—the BGP decision process selects one best route for all customers, precluding the "win-win" opportunity for large ISPs and their customers.

A single ISP *can* safely and effectively act alone.

An ISP can apply more flexible routing policies, without compromising global routing stability, while remaining backwards compatible with existing routers.

Global stability: Since some combinations of routing policies cause the global routing system to oscillate [11], our improvements in flexibility must be made judiciously. Fortunately, an ISP can safely advertise *any* route to neighbors that do not announce BGP routes to others [38]—stubs (i.e., ASes that do not provide transit service) fall in this category. Our analysis in the Appendix shows that the number of stub ASes is substantial: 84.1% of all ASes (22001 out of 26151) are stubs. For the six ISPs with more than 1000 customers, 60% of the customers are stub ASes; for the two largest ISPs (with over 2000 customers each), more than 80% of the customers are stubs.

Backwards compatibility: Although a "flag day" for upgrading BGP may not be possible, evolutionary changes can offer substantial improvements. At a high level, local routing policy operates as a "black-box" that, given a set of candidate routes as input, selects the best route for each prefix. Therefore, an ISP can change how the "black-box" works without modifying the protocol or requiring cooperation from other ASes. For example, previous work has shown how to move control-plane functionality to a small collection of servers that select BGP routes on behalf of the routers [9, 4, 33, 34]. In the rest of the paper, we show how to design a routing control platform that enables an ISP, acting alone, to realize many useful routing policies that are infeasible today.

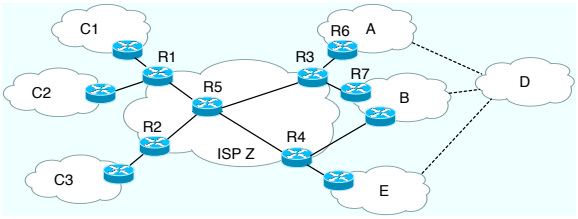

Figure 1: ISP Z has multiple interdomain routes to D

3 Intra-AS Routing Architecture

In this section, we present the design of the intra-AS routing architecture Morpheus relies on to enhance the flexibility of implementing routing policies beyond what is achievable today. We discuss three requirements of the routing architecture that are essential to achieve this goal. We then propose three corresponding changes to the way routes are disseminated or assigned within a single AS. These changes enable Morpheus servers to: (1) have complete visibility of all available routes, (2) assign any route through any egress link to any neighbor AS independently, and (3) assign egress point to every edge router independently. All the three routing-architecture changes are backwards compatible and are local to the ISP that deploys Morpheus.

3.1 Complete Visibility of BGP Routes

As discussed in Section 2, path diversity is the basis of policy flexibility. However, much of the path diversity of a tier-1 ISP remains unused as routers do not have complete visibility of BGP routes [32]. An edge router may learn multiple routes for the same destination prefix through eBGP sessions with neighbor ASes. However, the router can only select and propagate one best route per prefix to other routers in the AS. As a result, there are many routes visible to only one router in an AS. For example, in Figure 1 router R3 learns two routes to destination D but only propagates one (say, the one via R6) to R1. Then, R1 does not learn, and hence cannot use the other route (via R7), even if R1 preferred this route (e.g., if its customer C1 wants to circumvent some particular AS in the path via R6). Such loss of visibility gets even more pronounced in large networks due to the use of route reflectors [32]. Although propagating only one route helps limit control-plane overhead, it imposes significant limitations on flexibility.

Requirement 1: *An AS should have complete visibility of eBGP-learned routes to enable flexible routing policies.*

Morpheus uses a small collection of servers to select BGP routes on behalf of all the routers in the AS. Morpheus can obtain full visibility of all available

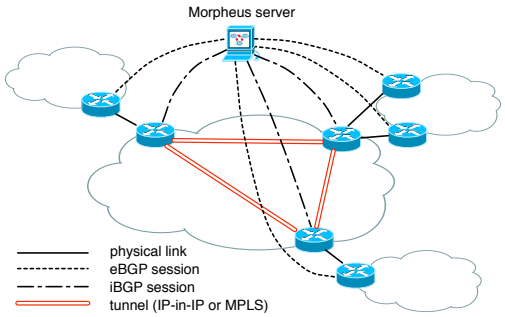

Figure 2: Morpheus routing architecture: Morpheus servers peer with neighboring domains via multi-hop BGP sessions; edge routers direct interdomain traffic through tunnels.

BGP routes through (multi-hop) eBGP sessions with the routers in neighboring ASes, as in the Routing Control Platform [9] [1]. Morpheus assigns BGP routes using internal BGP (iBGP) sessions between the servers and the routers for backwards compatibility. The Morpheus servers also ensure that the BGP routes propagated to eBGP neighbors are consistent with the routes assigned to the associated edge links.

3.2 Flexible Route Assignment

Today, even *with* complete visibility of alternative routes, a router cannot allow different customers to use different paths. In Figure 1, the two customers C1 and C2 connected to the same edge router R1 may want to use the two different paths through the same egress point R3 to reach D, respectively. To make such policy possible, the AS has to have the ability to (1) use available paths through any *egress link* (rather than *egress router*) flexibly, and (2) assign those routes *independently* to different neighbor ASes (whether connected at the same edge router or not).

Requirement 2: *An AS should be able to assign any route through any egress link to any neighbor AS independently.*

With full visibility of all eBGP learned routes, Morpheus can easily pick the best routes through any egress link for its customers and edge routers individually. Morpheus can disseminate potentially multiple routes per prefix to edge routers in several ways [2]. Since the edge 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 memory reduction on edge routers. Upon receiving these routes, edge routers can use the "virtual routing and forwarding (VRF)" feature commonly used for MPLS-VPNs to install different FIB entries for different customers [21].

3.3 Consistent Packet Forwarding

Care must be taken to ensure the aforementioned flexibility in route assignment does not introduce inconsistent forwarding and potential forwarding loops.

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. This rule ensures consistent forwarding decisions among the routers in the network. 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 [30].

Requirement 3: *An AS should be able to assign any egress point to every edge router independently (without causing forwarding loops).*

To achieve this goal, Morpheus relies on IP-in-IP or MPLS tunnels to direct traffic between edge routers, as shown in Figure 2. This design choice offers several important additional advantages. First, Morpheus can freely assign different BGP routes to different edge routers or links, 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 the Morpheus server 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; that is, Morpheus only assigns routes to the egde routers that connect the ISP to other networks. Fourth, tunneling gives greater flexibility in how the egress points are selected, for more flexible traffic engineering. Such *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.

[1] Alternatively full visibility of the routes can be obtained through BGP Monitoring Protocol (BMP) sessions [27], which is more scalable.

[2] This can be achieved by using the "route target" attributes commonly used with VRF in MPLS-VPN [21], or having multiple iBGP sessions between a Morpheus server and an edge router. Other options include using the BGP "add-paths" capability [36] or a new message dissemination protocol, which may be more efficient at the expense of backwards compatibility.

4 Server Software Architecture

In this section, we present the software architecture of the Morpheus server. To guide our design, we appeal to Multi-Criteria Decision Analysis (MCDA), a fast-growing area in decision theory. Formulating path selection as an MCDA problem leads us to separate *route classification* (tagging routes based on distinct policy objectives) from *route selection* (by computing a weighted sum of the tags). Since the inputs and outputs of Morpheus servers are just regular BGP update messages, and the tags and weights are purely internal to Morpheus, we achieve these gains in flexibility without sacrificing backwards compatibility.

4.1 Route Selection as an MCDA Problem

In today's ISP networks, specifying routing policies is intertwined with configuring "route-maps" that modify a small set of route attributes (such as local preference) to indirectly influence BGP's multi-stage decision process. Stepping back from the somewhat convoluted mechanisms in BGP, we need to articulate the problem the network operators are trying to solve. Abstractly, the ISP learns a set of routes for each destination prefix and needs to assign each edge router, or edge link, a single "best route" from that set. The right choice depends on several criteria, such as the business relationship with the next-hop AS, the perceived end-to-end performance of the forwarding path, the stability of the route, and so on. Even with complete visibility of the paths, and complete control over path assignment, the network operators cannot escape the need to reconcile the relative importance of these goals.

The challenges of weighing multiple, sometime conflicting criteria, also arises in many other disciplines. These problems are known in the decision-theory community as *multi-criteria decision analysis* (MCDA) problems [2]. MCDA provides a meaningful way to balance trade-offs between the various criteria. For BGP route selection, we can model the problem as follows:

Route-selection process (RSP) as an MCDA problem: *Given a set of available routes $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ for a prefix p, choose a best route r^* according to a set of criteria $\mathcal{C} = \{c_1, c_2, ..., c_k\}$.*

Example criteria include stability (where routes are rated based on their flapping history), performance (where routes are rated based on active measurements of the forwarding path, or reputation of the ASes along the AS path), and security (where routes are rated based on agreement with public registries or the routes announced in the past). In Morpheus, these ratings are computed by *classifier* modules that operate on the incoming BGP update messages, as shown in the left-hand side of Figure 3.

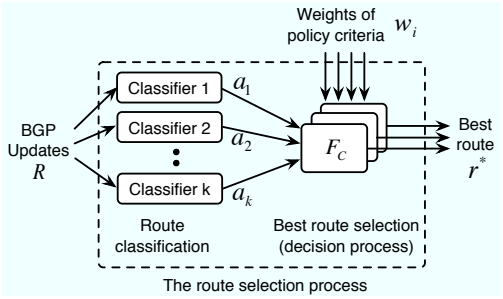

Figure 3: The MCDA model of the route selection process (RSP) problem

After identifying the set of k criteria $\mathcal{C} = \{c_1, c_2, ..., c_k\}$ of interest, network operators need to specify trade-offs amongst the criteria, i.e., a "decision function" $\mathcal{F}_\mathcal{C}$, as shown in the right-hand side of Figure 3. The decision-theory community has proposed a variety of methods for solving MCDA problems [2, 31, 18, 16]. One simple, yet powerful, technique is a *weighted sum*. For a route $r \in \mathcal{R}$, its weighted sum *score* is:

$$S(r) = \sum_{c_i \in \mathcal{C}} w_i \cdot a_i(r) \quad (1)$$

where w_i is the weight for criterion c_i in $\mathcal{C}$, and $a_i(r)$ is route r's numerical label generated by the classifier of c_i. For a prefix p, the decision function $\mathcal{F}_\mathcal{C}$ selects the alternative route with the highest final score as the best choice:

$$r^* = \mathcal{F}_\mathcal{C}(r) = \arg\max_{r \in \mathcal{R}(p)} S(r) \quad (2)$$

The weighted sum provides an expressive interface for balancing trade-offs between the k criteria (i.e., by configuring the weights). A weighted sum is simple to compute independently for each route, and to compare new routes to the current best route. In Morpheus, we envision supporting multiple such functions $\mathcal{F}_\mathcal{C}$'s (as shown in Figure 3), each with different weights, to satisfy customers with different priorities.

4.2 Policy Classifiers

Guided by the above model, we design the Morpheus server to have a set of *policy classifiers* before the *decision process*, as shown in Figure 4.

Tagging the routes: Each classifier takes in a route as input, examines the route according to a specific policy criterion, and generates a tag that is affixed to the route as metadata. For example, a business-relationship classifier may tag a route as "customer", "peer", or "provider";

a security classifier that detects suspicious routes (e.g., those being hijacked) may tag a route as "suspicious" or "unsuspicious" [13]; a stability module (using, for example, a route-flap damping algorithm to evaluate route stability) may tag a route with a route-flap damping penalty value [35].

Each policy classifier works independently and has its own tag space, obviating the need to overload the same attribute. As each policy classifier works independently, it is easy to extend the system with a new policy objective by adding a new classifier, without changing or affecting any existing ones. Furthermore, when a new module needs to be incorporated into the system, upgrades need only be applied to the Morpheus servers instead of all routers in the AS. These classifier-generated tags are purely local to Morpheus, and are never exported with BGP update messages; as such, using these tags does not require any changes to any routers.

By tagging the routes, rather than filtering or suppressing them, the decision process is guaranteed to have full visibility of all valid candidate routes (except those that are ill-formed or cannot be used under any circumstances, e.g., those with loops in their AS paths). This is in sharp contrast to the current BGP implementation in which all the routes for the same prefix may be filtered or suppressed (e.g., in the case of route-flap damping), leaving the decision process with no route to choose from.

Incorporating "side information": Many useful policy objectives require certain *side information*, including *external information* such the business-relationships with the neighbors, measurement data, or a registry of prefix ownership, or *internal states* such as a history of ASes that originated a prefix. However, there is no systematic mechanism to incorporate side information to routers today. Network operators have to either "hack" their BGP configurations in an indirect and clumsy way (e.g., reconfiguring filters and community attributes), or wait for software upgrades from router vendors (if the need for certain side information becomes compelling) and then upgrade a large number of routers.

The introduction of policy classifiers makes it easy to incorporate "side information" as each policy classifier can have access to custom external data sources containing the information needed to classify the routes. For example, the business-relationships classifier can have access to up-to-date information about the ISP's business relationships with neighboring ASes through a corresponding configuration file. A security classifier can have access to a registry of prefixes and their corresponding owners. A performance classifier can tag a route based on a reputation system (e.g., AS X is well known to have bad performance, circumvent it if possible), or real-time measurement results from a monitoring system.

Different classifiers can also maintain separate internal states. For instance, a stability classifier can maintain statistics about route announcement and withdrawal frequencies. A route security module that implements Pretty Good BGP (PGBGP)—a simple algorithm that can effectively detect BGP prefix and subprefix hijacks—can keep past history of BGP updates in the past h days (where h is a configurable parameter) [13].

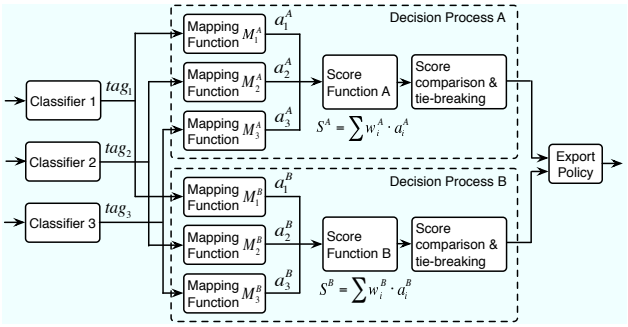

Figure 4: Each decision process consists of a mapping function and a score function. Different decision processes are configured with different mapping functions and/or score functions to realize different policies.

It is worth pointing out that network operators do not have to implement all the policy modules themselves. Instead, they can "plug and play" classifier modules developed by third parties. We believe that our open platform will foster the sharing of modules in the operations community and may also lead in the long run to the emergence of a market centered around these modules.

4.3 Configurable Decision Processes

Unlike the restrictive step-by-step decision process of the current BGP implementations, Morpheus' decision process enables network operators to make flexible tradeoffs between policy objectives. Furthermore, Morpheus supports the parallel execution of multiple decision processes to implement customer-specific decisions.

Flexible policy control: In Morpheus, each decision process consists of two steps: (1) mapping tags tag_i generated by different classifiers to numerical values a_i in a normalized space, and (2) calculating the score s for each route, as shown in Figure 4. In comparison to the MCDA model shown in Figure 3, the decision process of Morpheus contains an additional mapping function $\mathcal{M}_i$ between each classifier and the score function. This additional step has two major benefits.

First, the introduction of the mapping functions decouples the *generation* of tags (the job of the classifiers), and the *interpretation* of tags (the job of the mapping functions). In this way, each policy classifier can tag routes in its own tag space without worrying about the consis-

tency with other classifiers. This facilitates the implementation of classifiers by third parties. The mapping functions ensure that all tags are converted to the same numerical space (e.g., $[0, 100)$) to make the comparison between different policy criteria meaningful.

Second, since Morpheus instantiates one decision process for each routing policy (i.e., a specific set of weights of policy objectives), the introduction of mapping functions enables different policies to interpret the same tag *differently* (e.g., one policy may want to set a threshold for route stability and treat all routes with penalty values below the threshold as "equally stable", while another policy may want to always select the most stable route available), as each decision process has its own mapping functions for all the policy criteria. As shown in Figure 4, the same tag tag_1 can be mapped to different values a_1^A and a_1^B by two different mapping functions $\mathcal{M}_1^A$ and $\mathcal{M}_1^B$. Therefore, network operators can customize their policies through different configurations of the mapping functions.

Network operators configure weights w_i for all the policy criteria $c_i \in \mathcal{C}$ to realize their preferences and tradeoffs. With the w_i's configured by the operators, and the a_i's output by the mapping functions, the score function calculates the score for a new route, compares it with the scores of other routes for the same prefix, and picks the one with the highest score as the best route. In the case that multiple routes have the same highest score, the tie can be broke in different ways, e.g., using configurable rank-based tie-breakers (as discussed in Section 6), or simply using router IDs.

Intuitive configuration interface: Although network operators can configure the mapping functions and score-function directly, Morpheus also provides a configuration interface that can be used to help derive the appropriate mapping functions and weights in the score function. This interface is based on the analytic hierarchy process (AHP), a widely-used method in MCDA [25]. When this interface is used, network operators are asked to give their preferences on a set of pairwise comparisons amongst policy objectives (e.g., "performance vs. security", "security vs. business relationships", etc), using a numerical scale of 1 to 9, where 1 means two options are equally preferred, while 9 means absolute (extreme) preference. Based on these answers (a matrix of preferences), the AHP can derive the appropriate weights for all objectives in an automated fashion using a standard decision-theoretic methodology, as discussed in [25]. To derive a mapping function, operators can first divide the corresponding tag space into several discrete intervals. For example, the loss rate tag space $[0, 1]$ can be divided into four intervals: $[0, 0.03)$, $[0.03, 0.1)$, $[0.1, 0.2)$, $[0.2, 1]$ [3]. After the operators specify their preferences over the set of intervals, the AHP will derive the appropriate numerical value for each interval. All tags fall into the same interval will be mapped to the same numerical value. Appendix B described the AHP-based configuration interface in greater details.

Supporting multiple policies in parallel: Morpheus supports the simultaneous realization of multiple independent routing policies through the parallel execution of multiple decision processes, each selecting its own best route for the same prefix. In practice, we envision that an ISP would provide a limited set of independent routing policies as services to which customers may subscribe. For example, the ISP could offer different types of routes such as "the most stable routes through a customer", "the most secure routes through any neighbor," and "the lowest latency routes". Furthermore, Morpheus allows an ISP to configure customer-specific policies to select special routes that are requested by certain customers.

5 Example Policies

In this section, we present example routing policies, including several recent proposals from the research community [10, 17, 15], that Morpheus can support. We start with two example policies that a single ISP alone can realize using Morpheus. Then, we discuss two more example policies that require cooperation between ISPs. We show that Morpheus provides a backwards compatible platform for a single ISP, or a group of cooperating ISPs, to realize policies that are infeasible today.

5.1 Flexible Policies in a Single AS

In this subsection, we discuss two example policies that a single ISP can realize using Morpheus. The first policy strikes a balance between route stability and other policy objectives, while the second enables an ISP to select different routes for different customers.

Improving BGP stability: Unstable BGP routes can cause data traffic to experience serious performance degradation [15]. The well-known route-flap damping (RFD) mechanism [35] improves stability by suppressing routes that flap often, at the risk of compromising reachability when every route is unstable. It is well known that RFD can create black holes for data traffic if all routes to the same prefix are suppressed. In practice, ISPs often disable RFD because they prefer to have *some* route, however unstable, than no route at all. Therefore, if all routes are unstable, it is desirable to have a policy

[3] This step can be omitted if the tag space is already discrete (e.g., "customer", "peer", "provider" in the business relationship case).

that picks the most stable route, rather than leaving the destination unreachable.

With Morpheus, such policy can be easily realized by having a stability classifier tag routes with a penalty value computed based on their stability history (e.g., as proposed in [35]) without ever suppressing any unstable routes. Here we present one intuitive way to implement the stability classifier, which computes the penalty value based on a route's *up-time*, i.e., the elapsed time since a route announcement is received. Using the up-time of a route for predicting the future stability of the route is intuitive and, in fact, is part of the recently proposed Stable Route Selection (SRS) algorithm [10].

Computing a score for a route based on its up-time requires the stability classifier, upon reception of a new route announcement, to tag the route with its arrival time. The arrival time is then mapped by the corresponding mapping function to a number in the normalized numerical space (as mentioned in Section 4.3). Network operators can configure the score function to always select the most stable route by assigning a large weight to the stability objective, or they can strike a balance between stability and other objectives (e.g., business relationships) by adjusting the weights accordingly.

Different performance goals for different customers: As mentioned earlier, a large ISP has both enough path diversity and the incentive to offer different paths to different customers. For example, an ISP can conceivably offer three types of paths with different performance properties: paths with low latency, paths with high throughput, and paths selected by preferring customer routes (as is common practice today). A customer hosting a VoIP service or online games may prefer the low-latency paths, a customer hosting rich contents may prefer the high-throughput paths, and other customers may be satisfied with the normal paths (at a lower cost). Such customized path selection can be implemented by adding a latency classifier, a throughput classifier, and instantiating three decision processes corresponding to the three customer types—one that places a high weight on the latency objective, one that places a high weight on the throughput objective, and one that implements the current BGP decision process.

Determining path performance is possible through different mechanisms. For example, the latency of a path can be estimated based on the geographic location of the ISP and the downstream ASes in the path—an ISP in Mexico may prefer a path of North American ASes to reach a network in Canada over paths containing European or Asian ASes. As another example, a throughput classifier can be configured to avoid a list of ASes known to have low-bandwidth peering links with their adjacent ASes. Such low-overhead approaches of coarse classification granularity may be sufficient for most applications. If finer-grained performance information is desired, the ISP may choose to either use performance monitoring systems such as Keynote [14] or perform its own customized monitoring using, for example, active probing for a limited set of prefixes of particular interest to its customers.

5.2 Cooperation Across ASes

Although Morpheus is primarily designed as a backwards compatible system to enable a single ISP to realize flexible policies, once two or more ISPs have deployed Morpheus individually, they can also use it as a platform for realizing policies that require bilateral or multilateral cooperation. The implementations of such policies typically require that ISPs share information among each other. Morpheus can facilitate the exchange of such information among the ISPs in a backward-compatible fashion, by setting up control sessions amongst the Morpheus servers, obviating the need for router upgrades.

Mutually Controlled Routing with Independent ISPs (Wiser): Mahajan *et al.* proposed Wiser [17], a BGP extension that enables ISPs to jointly control routing in a mutually beneficial manner, even if each ISP acts according to its own interest. Wiser is intended to replace hot-potato routing in which an ISP selects the closest egress link to the downstream AS for forwarding transit traffic. Each ISP running Wiser uses its own criteria to compute a *Wiser cost* for each route, based on the local IGP cost of that route and the corresponding external costs announced by neighboring ASes. Mahajan *et al.* propose to add a step in the BGP decision process, after the local-preference comparison, to select routes based on their *Wiser cost*. Wiser can be easily supported in Morpheus by adding a new policy classifier implementing the Wiser-score calculation algorithm and tagging each route with a score equal to the Wiser cost.

Pre-computed Failover Paths (R-BGP): Kushman *et al.* recently proposed R-BGP, a BGP extension that avoids the transient loss of connectivity during routing protocol convergence. In R-BGP, an ISP announces pre-computed failover paths to the neighboring ASes so that when a path fails the temporary loss of connectivity can be avoided if the neighboring ASes promptly switch to the failover paths.

Morpheus provides a suitable platform to deploy R-BGP for two reasons. First, R-BGP requires modifications to routers to compute failover paths and to enable the propagation of multiple paths for the same prefix. With Morpheus, such computation and information exchange can be performed by the Morpheus servers without having to upgrade the routers. Second, Morpheus has better visibility of the available BGP paths than is

possible for any single individual router. Because Morpheus can perform the computation using more paths, the quality of the selected failover paths (measured based on how disjoint the alternate path is in comparison to the currently active path) can be improved, making R-BGP more effective.

6 Implementation

We have implemented a Morpheus prototype as an extension to the XORP software router platform [12]. We first highlight the major changes we made to XORP, then describe the four policy classifiers and the decision process we have implemented in greater detail.

6.1 Changes to XORP

We chose XORP as the base platform to implement our Morpheus prototype because its modular structure closely parallels the Morpheus software architecture. However, since XORP is designed to implement the standard BGP decision process operating at a single router, our prototype differs from XORP's BGP implementation in three key ways.

First, we implemented the weighted-sum-based decision process of Morpheus from scratch. It has the ability to select different routes for different edge routers/peers, and can simultaneously run multiple decision processes each having its own policy configuration.

Second, to demonstrate that a policy classifier is easy to implement and to evaluate the performance of different such classifiers in action, we implemented four policy classifiers performing classifications based on business relationships, latency, stability and security respectively. While these classifiers could, in principle, work in parallel, in our prototype we implemented them as new modules in the XORP message processing pipeline, as shown in Figure 5. Since the classifiers work independently, the ordering amongst them is not critical.

Third, we modified XORP's import and export-policy modules to bypass route-flap damping, and ensure export consistency between edge routers and the neighboring domains connected to them.

6.2 Policy Classifiers

In this section, we discuss in detail the four policy classifiers we have implemented thus far.

Business relationships: The business relationship classifier is linked to a configuration file that contains a table of {next-hop AS, business relationship} pairs. When a Morpheus server is started, it reads this file into memory. When a new route arrives, the classifier consults the table and assigns the route with the appropriate tag (e.g., "customer", "peer", or "provider").

Latency: Our latency classifier assumes there is a performance monitoring system (PMS) from which it periodically pulls real-time latency information about paths between an ingress point to an egress point, and from an egress link to a destination prefix. The retrieval of the performance information is handled by a background process and the pulling interval can be adjusted to reach a sweet spot between the freshness of the latency information and the communication overhead to the PMS.

The latency classifier generates two types of tags—the absolute latency and the relative latency, to serve different policy needs—some policies only care about the relative latency amongst alternative paths (e.g., "always choose the path with the lowest latency"), while others may be specific about absolute latency (e.g., "for all paths with latency less than 100 ms, choose the most stable one through a customer"). To generate both types of tags, the latency classifier internally keeps records of the current path latency t_{now} and the minimum observed path latency t_{min} for each (prefix, next-hop) pair. When a new route arrives, it is tagged with t_{now} in milliseconds as the absolute latency, and t_{now}/t_{min} as the relative latency.

Stability: Our stability classifier implements the same penalty function as route flap damping does [35]. However, instead of suppressing routes when their penalties exceed a threshold, our stability module simply tags each route with a penalty value.

Security: Our security classifier implements Pretty Good BGP (PGBGP) [13], a simple yet effective heuristic algorithm that identifies bogus routes based on a history of {prefix, origin AS} pairs. A route is tagged as "suspicious" if a route's AS path does not match the history of the last h days (where h is a configurable parameter); or as "unsuspicious" otherwise. This classifier is ported by the author of PGBGP from his original implementation, with a few interface changes. This demonstrates that the design of Morpheus is friendly to third-party modules.

Amongst the four classifiers, three of them (except the business-relationships classifier) are required to "re-tag" previously tagged routes when certain conditions are met. For example, the latency classifier needs to re-tag a route if the change in path latency exceeds a certain threshold. The stability classifier needs to re-tag a route when the decay of its penalty value exceeds a preset limit. The PGBGP algorithm also requires to re-tag a "suspicious" route as "unsuspicious" if it is not withdrawn after 24 hours. In all such cases, a configurable minimum re-tagging interval can be set to prevent unde-

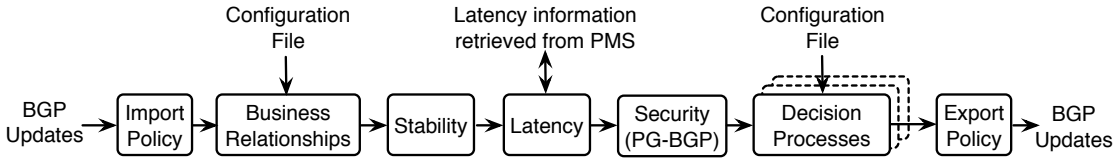

Figure 5: Morpheus prototype implemented as an extension to XORP

sirable flapping effect. (The 24-hour interval in the PG-BGP case is long enough, so no additional constraint is needed.)

6.3 Decision Processes

We implemented the decision process with four mapping functions for the four classifiers, and a weighted-sum score function, as described in Section 4.3. Our implementation assumes the mapping functions and the score functions are specified in configuration files ahead of time. When a new route arrives, a decision process only computes the score for this new route, without recalculating the scores for all previously received routes for the same prefix. In fact, the decision process only compares the new route's final score with the current *highest* score of that prefix. On the other hand, when the current best route is withdrawn, the decision process compares the scores of all remaining routes and picks the one with the highest score as the new best route.

It is possible that more than one route receives the same score. To select a single best route for each peer/edge router in that case, Morpheus currently supports two types of tie-breaking mechanisms—ranking of egress points, and router ID. In the rank-based tie-breaking scheme, each edge router is assigned with a fixed (but configurable) ranking of all egress points. This ranking may reflect geographic distance or the typical IGP distances and link capacities between each pair of ingress/egress points. By decoupling changes in the IGP distances from the decision processes, the fixed-ranking scheme avoids the problems associated with hot-potato routing [30] and gives the ISP additional control over the flow of traffic (e.g., decrease an ingress point's ranking of a particular egress point, if a link gets overloaded by the traffic from the ingress point to that egress point). 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.

7 Evaluation

In this section, we evaluate the performance and scalability of Morpheus using our XORP-based prototype. Specifically, we answer four questions:

1. *What is the performance of Morpheus' policy classifiers and its score-based decision process? (What is the performance bottleneck, if any?)* We find that the Morpheus classifiers and decision process work efficiently. The average decision time of Morpheus is only 20% of the average time the standard BGP decision process takes, when there are 20 routes per prefix.

2. *Can Morpheus keep up with the rate of BGP update messages in a large ISPs?* Our unoptimized prototype is able to achieve a sustained throughput of 890 updates/s, while the aggregated update arrival rate of a large tier-1 ISP is typically no larger than 600 updates/s [34].

3. *How many different policies (i.e., decision process instances) can Morpheus support efficiently?* Our experimental results show that our prototype can support 40 concurrent decision processes while achieving a sustainable throughput of 740 updates/s.

4. *What is the memory requirement for Morpheus servers?* We find our Morpheus prototype only consumes 10% more memory than the XORP BGP implementation it is based on.

7.1 Testbed

We conduct our experiments on a three-node testbed, consisting of an update generator, a Morpheus server, and an update receiver, interconnected through a switch. For a realistic evaluation, the route generator replays the RIB dump from RouteViews on April 17, 2007 [24] to the Morpheus server.

The evaluations were performed with the Morpheus server and the update generator running on 3.2GHz Intel Pentium-4 platforms with 3.6GB of memory. We run the update receiver on a 2.8GHz Pentium-4 platform with 1GB of memory. The three machines each has one Gigabit Ethernet card and are connected through a Gigabit switch. They all run Linux 2.6.11 kernel.

7.2 Processing Time

To evaluate the performance of Morpheus' policy classifiers and decision process, we conduct white-box testing by instrumenting the classifier functions and the decision process, and measuring the time they take to process a route. To highlight the performance difference

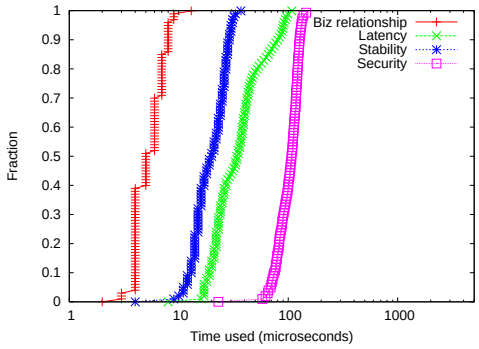

Figure 6: Classification time: time taken by the classifiers to tag a route

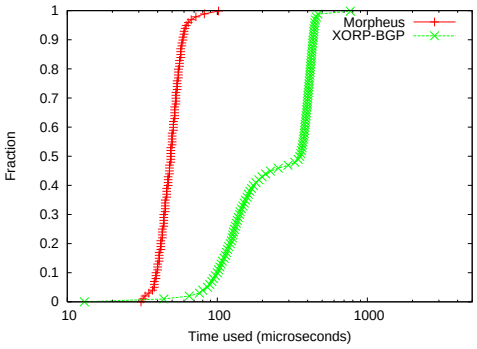

Figure 8: Decision time: comparison between Morpheus and XORP-BGP (20 routes per prefix)

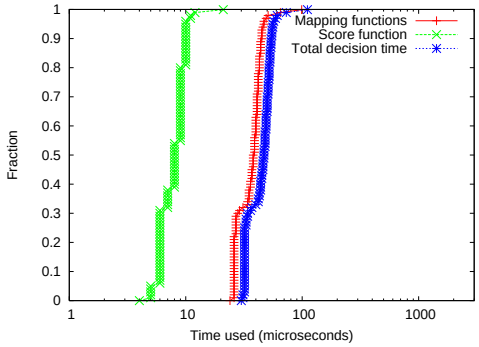

Figure 7: Decision time: time taken by the mapping functions and the score function, and the total decision time (1 route per prefix)

introduced by the Morpheus design, we also compare Morpheus' decision time with two reference implementations in XORP: the standard BGP decision process and a modified BGP decision process which uses rank-based tie-breaking (similar to what Morpheus uses) after the multi-exit discriminator (MED) comparison step. In each processing-time experiment, the update generator sends 100,000 updates to the Morpheus server.

Classification time: We first measure the time each policy classifier takes to tag a route. In this experiment, the business-relationship classifier reads in a table of 2000 {AS number, business relationship} pairs. The latency classifiers is supplied with a static table of path latency data. We believe the results we get should be comparable to the scenario in which Morpheus gets this information from a monitoring system, because the measurement results will be pre-fetched by a background process and cached. From the CDF of the tagging time shown in Figure 6, we see that the business-relationship classifier takes only about 5 microseconds to tag a route. The stability classifier takes about 20 microseconds on average, while the delay classifier takes about 33 microseconds. The most complex classifier—the security classifier which implements the PGBGP algorithm, takes 103 microseconds on average.

Decision time (one route per prefix): We then benchmark the time taken by the decision process to calculate the final score for a route. Figure 7 shows the CDFs of the two components of the decision time—the mapping functions (one for each classifier) and the score function, as well as the total time. As we expected, the score function runs very quickly, taking only 8 microseconds on average. The four mapping functions take 37 microseconds in total. The total decision time is about 45 microseconds on average. In this experiment, the update generator only sends one update per prefix to the Morpheus server, so there is no tie-breaking involved in our measurements.

Decision time (multiple alternative routes per prefix): In the next experiment, we compare the decision time of Morpheus and the out-of-box BGP implementation of XORP (XORP-BGP), when each prefix has multiple alternative routes. We configure both Morpheus and XORP-BGP to receive 20 identical (except for router IDs) routes per prefix from the update generator. To make fair comparison, we configure Morpheus to use router ID to break ties. From Figure 8 we can see Morpheus takes about 54 microseconds on average to select a best route, whereas XORP-BGP takes an average time of 279 microseconds.

It is not surprising to see that Morpheus takes much less time than XORP-BGP in selecting best route when the number of alternative routes is large, because regardless of the number of alternative routes per prefix, Morpheus only needs to compute one score when a new route arrives, whereas XORP-BGP has to compare the pool of alternative routes for the same prefix all together through the step-by-step comparisons in the BGP decision process. This also explains why the decision time of Morpheus has small variation, while XORP-BGP's decision time varies significantly, ranging from less than 100 mi-

croseconds (when there is only a small number of alternative routes for a prefix) to over 500 microseconds (when the number becomes large).

Table 1: Processing time of the rank-based tie-breaker

	10 routes/prefix	20 routes/prefix
10 edge routers	83 μs	175 μs
20 edge routers	138 μs	309 μs

Time to perform rank-based tie-breaking: Finally we measure the time Morpheus takes to perform rank-based tie-breaking when multiple alternative routes have the same score. Without any knowledge about the how often and how many routes will end up having the same score, we study two cases in our experiments: the *random case* and the *worst case*. In the random case, we assign every alternative route with a random integer score uniformly selected between 0 and 100. In the worst case, we let all alternative routes per prefix have the same score. We run eight test cases: random case/worst case with 10/20 edge routers and with 10/20 routes per prefix. Since in the four random cases, there is little possibility ($C_{20}^2 \cdot 0.01^2 = 0.019$) that two routes will have the same final score, leaving the rank-based tie-breaker almost never used, we list only the average tie-breaking time of the four worst cases in Table 1. As we can see, if *all* alternative routes happen to have the same score, the rank-based tie-breaking step will become the performance bottleneck of Morpheus' decision process, even in the modest case of 10 routes/prefix with 10 edge routers. However, such worst case scenario is not likely to happen very often in reality, especially when the number of alternative routes is relatively large.

7.3 Throughput

To determine the update processing throughput Morpheus can achieve, we use the following network model of a large tier-1 ISP from a prior study [34]. We assume a large ISP has 40 Point-of-Presence (POPs), each of which contains one Morpheus server. Each Morpheus server has 240 eBGP sessions with customers, and 15 iBGP sessions with edge routers. It also keeps 40 sessions with other Morpheus servers, through which it learns every route other Morpheus server receives. We assume the ISP (and each of its Morpheus servers) receives 20 routes per prefix (as shown in Section 2). Since each Morpheus server selects routes for the edge routers located the same POP, in the experiments we assume it applies the same ranking of egress points for all its 15 edge routers, while different Morpheus servers still have different rankings.

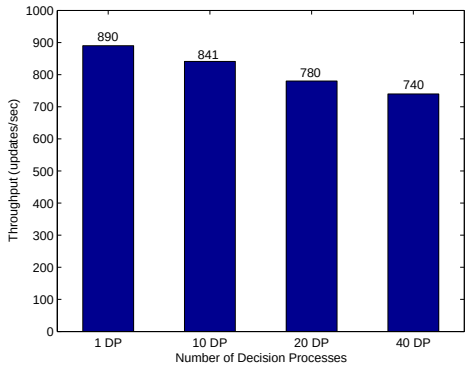

Figure 9: Throughput achieved by Morpheus with different number of decision processes

In each throughput experiment, the update generator maintains 20 sessions with the Morpheus server and sends 20 update messages of each route, one per session. The Morpheus server maintains 295 sessions (240 eBGP sessions, 15 iBGP sessions and 40 sessions to other Morpheus servers) with the update receiver. By sending multiple routes with the same attributes to the Morpheus server, we measure the *worst case* throughput Morpheus can achieve, because all the routes will have the same score and hence the rank-based tie-breaking step is always performed. Every throughput experiment runs for a 15-minutes period, and the update generator sends updates at the fastest rate it can get.

Figure 9 compares the throughput achieved by Morpheus configured with different number of decision processes. When Morpheus only runs one decision process, it achieves a sustained throughput of 890 updates/s. As the number of decision processes increases to 10, 20 and 40, the achieved throughput decreases slowly to 841, 780 and 740 updates/s, respectively. When we increase the number of decision processes, we assume each customer still subscribes to only one of them (i.e., only receives one route per prefix). As such, the total number of updates the Morpheus sends to the update receiver does not increase.

We are satisfied with the throughput our unoptimized prototype achieves, as a large tier-1 ISP usually receives less than 600 updates/s (95 percentile) [34]. The slow decrease of throughput as the number of decision processes increases also demonstrates Morpheus' score-based decision process design can scale to a large number of different policies.

7.4 Memory Requirement

When we compare the memory consumption of Morpheus with XORP-BGP, we find XORP-BGP consumes 970 MB of memory when loaded with five routes per prefix, as its implementation stores multiple copies of each route. Therefore, neither the out-of-box XORP-BGP nor our XORP-based Morpheus prototype were able to load 20 full BGP routing tables with 3.6 GB of memory. However, the memory footprint of our Morpheus prototype is only 10% larger than that of XORP-BGP, which is mainly used by the classifiers (largely by the security classifier) and used to store metadata of routes (tags, scores, etc.). Given that other BGP implementations consume significantly less memory (e.g., openbgpd only takes 270 MB to store 20 full BGP routing tables [34]. Our experiment on Quagga [22] observes a memory footprint of 550 MB with 20 full BGP routing tables.), we believe a Morpheus prototype based on a BGP implementation with better memory efficiency will not impose any memory pressure on a reasonably provisioned server. We also note that, unlike router memory, memory for regular servers is cheap and easy to install.

8 Related Work

Previous work proposes to raise the level of abstraction of BGP policy configuration through network-wide, vendor-neutral specification languages [1, 3]. However, we believe new languages alone are not sufficient to make policy configuration more flexible, because today's intra-AS routing architecture and the current BGP decision process both introduce peculiar constraints on the set of policies that can be realized. In this paper, we take a fresh approach to the problem by modeling the route selection problem within an AS in the MCDA framework and designing a system guided by that model.

Several recent studies on the Routing Control Platform (RCP) [9] advocate moving the BGP control plane of a single AS to a small set of servers that select routes on behalf of the routers [4, 33, 34]. The prototype systems in [4] and [34] demonstrate that a logically-centralized control plane running on commodity hardware can be scalable, reliable, and fast enough to drive BGP routing decisions in a large ISP backbone. However, system in [4] simply mimics the standard BGP decision process, without expanding the space of realizable policies. While [33] and [34] support more flexible alternatives to today's hot-potato routing, these systems do not create an extensible framework for specifying and realizing flexible policies, or explore how to support trade-offs between policy objectives or multiple policies simultaneously. These are the main contributions of our Morpheus design.

9 Conclusion

This paper presents the design, implementation and evaluation of Morpheus, a routing control platform that enables a single ISP to realize many useful routing policies that are infeasible today, without changing its routers or coordinating with other domains. Inspired by Multi-Criteria Decision Analysis (MCDA), the design of the Morpheus server separates route classification from route selection, which enables network operators to easily define new policy objectives, implement independent objective classifiers, and make flexible trade-offs between objectives. Morpheus allows large ISPs to capitalize on their path diversity and provide customer-specific routes as a value-added service. Our experiments show that Morpheus can support a large number of different policies simultaneously while handling the high rate of BGP updates experienced in large ISPs.

Most policy objectives can be expressed in terms of tags or scores for individual routes. A notable exception is traffic engineering (TE), since the total traffic on each link in the network depends on the mixture of traffic from many interdomain paths. Today, network operators perform TE by tuning the IGP link weights and BGP routing policies to move traffic away from congested links. With Morpheus, the network operators can also configure the egress-point rankings to manipulate the flow of traffic. In addition, even if some customers subscribe to customized routes, the remaining customers use whatever paths the ISP selects as the "default". Controlling the route-selection process for the default customers give the ISP substantial leeway to perform TE. As such, we believe that providing greater flexibility in path selection is compatible with effective traffic engineering. We believe that exploring these issues in greater depth is a promising avenue for future research.

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] 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.

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

[6] Inferred AS Relationships Dataset from CAIDA. http://as-rank.caida.org/data.
[7] Cisco Optimized Edge Routing. http://www.cisco.com/en/US/products/ps6628/products_ios_protocol_option_home.html.
[8] N. Duffield, K. Gopalan, M. R. Hines, A. Shaikh, and J. E. van der Merwe. Measurement informed route selection. In *Proc. Passive and Active Measurement Conference (Extended Abstract)*, 2007.
[9] 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.
[10] P. B. Godfrey, M. Caesar, I. Haken, S. Shenker, and I. Stoica. Stable Internet route selection. NANOG40, June 2007.
[11] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Networking*, 10(1):232–243, 2002.
[12] 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.
[13] J. Karlin, S. Forrest, and J. Rexford. Pretty good BGP: Improving BGP by cautiously adopting routes. In *Proc. International Conference on Network Protocols (ICNP)*, November 2006.
[14] Keynote Systems. http://www.keynote.com.
[15] N. Kushman, S. Kandula, D. Katabi, and B. Maggs. R-BGP: Staying connected in a connected world. In *Proc. Networked Systems Design and Implementation*, April 2007.
[16] I. Linkov, A. Varghese, S. Jamil, T. Seager, G. Kiker, and T. Bridges. Multi-criteria decision analysis: A framework for structuring environmental policy decisions at contaminated sites. In *Proc. US-IALE 19th Annual Symposium*, March 2004.
[17] R. Mahajan, D. Wetherall, and T. Anderson. Mutually controlled routing with independent ISPs. In *Proc. Networked Systems Design and Implementation*, 2007.
[18] J. Malczewski. *GIS and Multicriteria Decision Analysis*. Wiley, 1999.
[19] W. Muhlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-topology model that captures route diversity. In *Proc. ACM SIGCOMM*, 2006.
[20] Discussion on NANOG mailing list. http://www.merit.edu/mail.archives/nanog/2007-04/msg00502.html.
[21] I. Pepelnjak and J. Guichard. *MPLS and VPN Architectures*. Cisco Press, 2000.
[22] Quagga Routing Suite. http://www.quagga.net.
[23] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC 4271, January 2006.
[24] The routeviews project. www.routeviews.org.
[25] T. L. Saaty. *The Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process, Vol. VI, AHP Series*. RWS Publications, 2000.
[26] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proc. ACM SIGCOMM*, 1999.
[27] J. Scudder. BGP monitoring protocol. Internet Draft draft-scudder-bmp-00, Auguest 2005.
[28] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proc. ACM SIGCOMM*, 2003.
[29] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proc. SPIE ITCom*, 2001.
[30] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, June 2004.
[31] E. Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study*. Springer, 2004.
[32] S. Uhlig and S. Tandel. Quantifying the impact of route-reflection on BGP routes diversity inside a tier-1 network. In *Proc. IFIP NETWORKING*, 2006.
[33] 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 (INM)*, September 2006.
[34] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. Snoeren, and J. van der Merwe. Wresting control from BGP: Scalable fine-grained route control. *Proc. USENIX*, 2007.
[35] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping. RFC 2439, November 1998.
[36] D. Walton, A. Retana, and E. Chen. Advertisement of multiple paths in BGP. Internet Draft draft-walton-bgp-add-paths-05, March 2006.
[37] 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 (HotNets)*, November 2006.
[38] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *Proc. ACM SIGCOMM*, September 2006.

APPENDIX

A Path Diversity

We analyze the number of stub-AS customers large ISPs typically have. We study a list 10 well known tier-1 ISPs [19], as shown in Table 2.

Table 2: The numbers of stub customers of some tier-1 ISPs

AS number	701	7018	174	1239	3356
Total customers	2634	2053	1667	1651	1425
Stub customers (%)	84.4%	86.1%	66.9%	78.9%	60.0%
AS number	209	3549	2914	3561	5511
Total customers	1233	924	460	449	131
Stub customers (%)	86.7%	57.8%	48.9%	72.8%	40.5%

We first count the total numbers of their *directly connected* customers using AS relationships inferred by CAIDA [6] on August 6, 2007. We then distinguish stub and non-stub customers by analyzing the global routing table snapshot taken by RouteViews [24] on the same day. We found that the 10 tier-1 ISPs have 8852 customers altogether (note that a customer

can be multi-homed to more than one of these ISPs), among which about 80% (7062) customers are stub ASes. Table 2 summarizes the break-down of the 10 ISPs. The top two ISPs (with over 2000 customers each) both have over 84% stub customers, and the top six ISPs (with over 1000 customers each) all have over 60% stub customers. Therefore, even with the conservative stability requirement, the majority of tier-1 ISPs' customers can still benefit from flexible policies enabled by Morpheus.

B AHP-based Configuration Interface

Besides allowing network operators to configure the mapping and score functions in a decision process directly, Morpheus also provides an analytic hierarchy process (AHP)-based configuration interface that helps operators derive the appropriate mapping functions and weights of the score function. We found the analytic hierarchy process (AHP) an attractive complement to the weighted sum method for two reasons: (1) like weighted sum, it uses a numerical score calculated in a weighted sum format to decide the best choice amongst a set of alternatives, (2) it provides a simple and intuitive interface to the network operators where they only need to specify their preferences on criteria in $\mathcal{C}$ using pair-wise comparisons (e.g., "strongly prefer performance over security", "weakly prefer security over relationships", etc.), and the mapping functions and weights of the score function can be derived automatically.

We first describe how AHP works, in the context of the RSP problem. We then show how AHP can be adapted to derive the mapping and score functions of the decision processes in Morpheus.

B.1 How AHP Works

The generic AHP algorithm [25, 2] can be applied in the following five steps to solve the RSP problem.

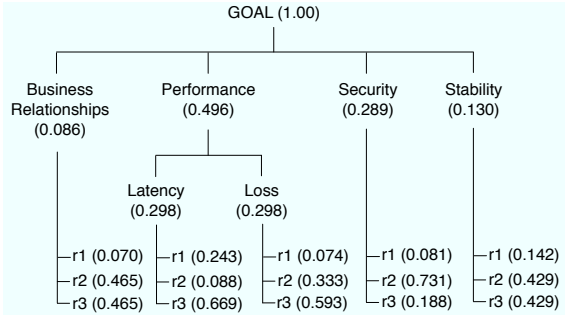

Figure 10: An example of applying Analytic Hierarchy Process (AHP) directly to solve the route selection process (RSP) problem (the offline algorithm). r1, r2 and r3 are three alternative routes to a prefix.

1. **Decompose:** Network operators identify the hierarchy of policy objectives (criteria). For example, in RSP the alternatives are $\mathcal{R}$, and one possible hierarchy of the criteria $\mathcal{C}$ is shown in Figure 10.

Table 3: An example of a matrix of preferences of business relationships (BR), performance (PER), security (SEC) and stability (STA)

Goal	BR	PER	SEC	STA
BR	1	1/5	1/3	1/2
PER	5	1	2	4
SEC	3	1/2	1	3
STA	2	1/4	1/3	1

2. **Specify preference on criteria:** Network operators specify their preferences on each pair of criteria (at the same level of hierarchy) in the form of a fraction between 1/9 and 9. When comparing criteria *p* to *q*, 1 means *p* and *q* are equally preferred, 3 means weak preference for *p* over *q*, 5 means strong preference, 7 means demonstrated (very strong) preference, 9 means absolute (extreme) preference, and the inverse values 1/3, 1/5, 1/7 and 1/9 are used in the reverse order of the comparison (*q* vs. *p*). Intermediate values (2, 4, 6, 8) may be used when compromise is in order. These pair-wise comparisons result in one or more matrices of preferences. For example, from the matrix in Table 3, we know that the network operator strongly prefer performance over business relationships, weakly prefer security over business relationships, etc.

3. **Specify preference on alternatives:** Network operators similarly specify their preference on each pair of alternative routes for each criterion at the bottom level of the hierarchy. For example, if there are three available routes in $\mathcal{R}$ for a prefix p, then the network operator need to do pair-wise comparisons of the three routes according to every criterion, i.e., business relationships, security, delay, loss and stability. Clearly, it is not practical to require network operators to compare routes manually, and therefore this algorithm only works in an *offline* fashion in its original format. In Section B.2, we describe how AHP can be adapted to derive the mapping functions which work in an *online* fashion.

4. **Derive weights for criteria and alternatives:** Given each matrix of preferences from Step 2 or 3, AHP derives weights for each criterion or each alternative in that matrix in a systematic way. For example, the preferences matrix in Table 3 results in the set of weights in the top level of hierarchy in Figure 10 [4]. The weights of all criteria sum up to 1. The weights of sub-criteria under a criterion sum up to the weight of that criterion. The weights of alternatives under each criterion sum up to 1.

5. **Calculate final score:** A final score is calculated for each alternative as a weighted sum of its weights for all the (sub)criterion at the bottom level of the hierarchy. For example, for a route r, its final score $S(r)$ is:

[4] AHP also automatically minimizes the inconsistency in the matrix of preferences (if any) while deriving the weights [25].

$$S(r) = \sum_{c_i \in \mathcal{C}} w_i \cdot a_i(r) \quad (3)$$

where $\mathcal{C}$ is the set of criterion at the bottom level of the hierarchy, $a_i(r)$ is r's weight for criterion c_i, and w_i is criterion c_i's weight amongst all the criteria. For a prefix p, the alternative route with the highest final score is considered the best choice.

B.2 AHP as a Configuration Interface

Score function: Since AHP uses similar weighted-sum score function as Morpheus does, Morpheus can derive the score function weights for all criteria by directly applying the steps 1, 2, 4 of the AHP algorithm described in Section B.1.

Mapping function: The mapping function case is less straightforward. As we mentioned in Section B.1, directly applying AHP in a mapping function can only work in an offline fashion since it requires pair-wise comparisons amongst all alternative routes. The key idea of making the offline AHP algorithm work online is the observation that when routes are compared according to a specific criterion in Morpheus, it is their corresponding tags of that criterion that are being compared. For instance, in business relationship, we compare one route's "customer" tag to another route's "peer" tag; in latency, we compare one route's "110ms" tag to another route's "200ms" tag, etc.

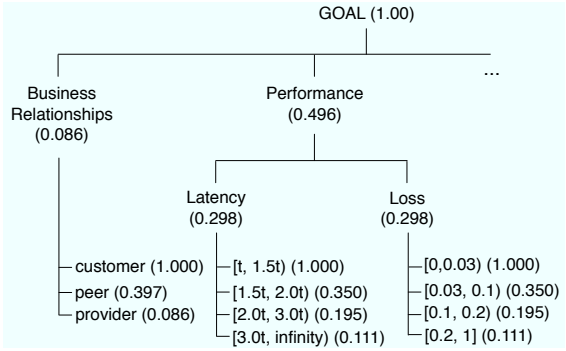

Figure 11: An example of the online AHP algorithm, highlighting the differences from the offline algorithm example in Figure 10. (For clarity, only part of the hierarchy is shown.)

Therefore, if we divide each criterion's attribute range into discrete intervals, network operators can do pair-wise comparisons to the set of intervals offline. For example, for business relationships, "customer", "peer" and "provider" naturally become the three attribute intervals. For criteria that originally have continuous attribute value ranges, such as latency and loss rate, discrete intervals can be divided with desirable granularity, e.g., $[t,\ 1,5t)$, $[1.5t,\ 2.0t)$, $[2.0t,\ 3.0t)$ and $[3.0t,\ \infty)$ for latency, where t is the observed minimum latency to reach a prefix. By having network operators compare attribute intervals in pairs offline, AHP can derive the numerical labels (weights) for each interval accordingly. This makes the online algorithm of the mapping function very simple: when a new route arrives, map its tag into the correct interval, and assign it with the corresponding numerical label. Figure 11 highlights the differences of the online algorithm from the offline algorithm shown in Figure 10.