Design for Configurability: Rethinking Interdomain Routing Policies from the Ground Up

Yi Wang, Ioannis Avramopoulos, *Member, IEEE* and Jennifer Rexford, *Senior Member, IEEE*

***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 configure, making many useful policies infeasible. In this paper, we present Morpheus, a routing control platform that is *designed for configurability*. Morpheus enables a single ISP to realize a much broader range of routing policies without requiring changes to the underlying routers or collaboration with other domains. Morpheus allows network operators to: (1) make flexible trade-offs between policy objectives through a weighted-sum based decision process, (2) realize customer-specific policies by supporting multiple route-selection processes in parallel, and allowing customers to influence the decision processes, and (3) configure the decision processes through a simple and intuitive configuration interface based on the Analytic Hierarchy Process, a decision-theoretic technique for balancing conflicting objectives. We also present the design, implementation, and evaluation of Morpheus as an extension to the XORP software router.**

***Index Terms*—BGP, interdomain routing, policy, configuration, analytic hierarchy process (AHP)**

I. Introduction

Internet Service Providers (ISPs) use interdomain routing policies to achieve many different network management goals, such as implementing business relationships with neighboring domains, providing good end-to-end performance to customers, improving the scalability of routing protocols, and protecting the network from attacks [8]. However, the *configurability* of ISP networks, i.e., the degree to which networks can be *customized* to implement routing policies, is limited because of the unnatural restrictions that BGP, the interdomain routing protocol of Internet, imposes on the way ISPs select routes.

BGP was designed when the Internet consisted of a small number of autonomous systems (ASes). Given the very limited path diversity within the small set of ASes, there was little need for a route selection process that supports configuration of flexible routing policies. However, as the Internet started to grow and path diversity increased, network operators started to demand more flexibility to configure more complex policies. The response from the vendors and standards communities was an incremental "patchwork" of backward compatible features to add attributes and steps to the BGP decision process [1]. (For example, AS_PATH was introduced in BGP-2, NEXT_HOP was introduced in BGP-3, and LOCAL_PREF was introduced in BGP-4.) The outcome was a decision process that is counter-intuitive and notoriously hard to configure. Today, despite the rich path diversity available to large ISPs, configurability is limited by restrictions imposed by virtually every aspect of policy configuration such as the routing architecture, the BGP software implementation, and its configuration interface.

For instance, each BGP 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. Within each router, the standard BGP implementation selects routes only based on the attributes of the BGP updates, falling short of realizing routing policies that, for example, require using outside measurement data. Finally, 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. This makes policies that strike a trade-off between different policy objectives hard to realize. For example, an AS cannot realize the following simple policy: "*If all routes are 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).*"

Stepping back, we ask the question: "Starting from a clean slate, how can we *design for configurability*?" That is, instead of seeking the best way to configure the existing system, we design a new system with configurability as a first-order goal. To make the new system practically adoptable, we focus on solutions that do not require cooperation between domains. Since ISPs are often business competitors, cooperation among them has proved notoriously difficult in practice. This constraint essentially prevents changes to the *interdomain* routing protocol that require collaboration of multiple domains. Fortunately, such changes are not necessary—large ISPs have a lot of path diversity, and can safely and effectively "act alone" in applying many flexible routing policies.

Y. Wang and J. Rexford are with the Department of Computer Science, Princeton University, Princeton, NJ 08540, USA (e-mail: yiwang@cs.princeton.edu, jrex@cs.princeton.edu).

I. Avramopoulos is with Deutsche Telekom Laboratories, Ernst-Reuter-Platz 7, 10587 Berlin, Germany (email: ioannis.avramopoulos@telekom.de).

To design for configurability, we consider the following **route selection problem** an ISP faces: *Given a set of available routes* $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ *for a prefix* p*, choose a best route* r^* *for each router according to a set of criteria* $\mathcal{C} = \{c_1, c_2, ..., c_k\}$. The set of criteria (i.e., policy objectives) includes route characteristics such as stability, security, and performance. These criteria may be conflicting in the sense that no route is the best with respect to all criteria simultaneously. Therefore, to design for configurability, the routing system must ensure that the network administrator has the flexibility to make arbitrary trade-offs among the criteria. Designing for configurability also means that the set of routes should be as large as possible. Our solution to the route selection problem is a system that we call *Morpheus* as it gives ISPs the power to "shape" their routing policies. Morpheus relies on the following system components:

- A **routing architecture** that is responsible for (1) learning the "inputs" and (2) disseminating the "outputs" of the route selection problem. The routing architecture allows a set of *Morpheus servers* to choose the best routes from the set $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ of *all* routes available to the AS, and ensures that the servers can assign any route in $\mathcal{R}$ *independently* to each neighbor without restrictions.
- A **server software architecture** giving the network operators the ability to make trade-offs among the criteria $\{c_1, c_2, ..., c_k\}$. It includes a set of *policy classifiers* and one or more *decision processes*. Each classifier tags routes with criteria-specific labels. The decision process computes a cumulative score as a weighted sum of the labels for each route and picks the route with the highest score. To pick potentially different routes for different neighbor networks (as supported by the routing architecture), multiple decision processes (possibly one per neighbor) can run in parallel.
- A **configuration interface** through which network operators can configure the decision processes. The straightforward method for a network operator to configure a decision process is to directly specify a weight for each criterion. However, without a systematic procedure for determining what the weights should be, this method would be error prone. Morpheus provides such a systematic procedure based on the Analytic Hierarchy Process (AHP) [28], which derives the appropriate weights based on operator's preferences on the policy objectives.

We have implemented Morpheus as a routing control platform consisting of a small number of servers that select BGP routes in a logically centralized way. Previous work on centralized routing platforms [7, 36, 35] has demonstrated that they can be made scalable and reliable enough for deployment in large ISP networks without sacrificing backwards compatibility. However, the previous work mainly focused on the *feasibility* of such logically centralized system, stopping short of addressing the poor configurability of BGP policies. In particular, the previous work did not identify the necessary supports for configurability from the routing architecture and proposed only limited improvements in the BGP decision process and its configuration interface.

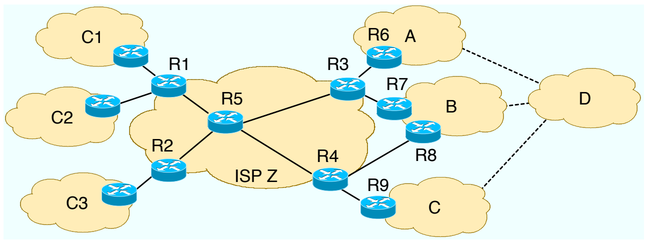


Fig. 1. ISP Z has multiple interdomain routes to D.

The rest of the paper is organized as follows. In Section II, we argue that more flexibility in interdomain routing decisions can translate into significant benefits for the ISPs without risking the stability of the routing system. In Section III, we identify the necessary changes to the current routing architecture in order to support flexible policies. We present the software architecture of the Morpheus server in Section IV, and give examples on how to configure routing policies through its AHP-based configuration interface in Section V. Section VI presents the evaluation of the Morpheus server as an extension to the XORP software router [17] and demonstrates that the gain in flexible policies does not come at the expense of scalability and efficiency. Finally, we present related work in Section VII and conclude in Section VIII.

II. 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.

A. Large ISPs Have Rich Path Diversity

Large ISPs that offer transit service usually connect to many neighboring ASes, often in multiple locations [23, 22]. For example, ISP Z 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 ISPs) have ten or more unique AS paths for certain destinations [23]. 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 [24]. A detailed study of an individual ISP reported an average of 20 router-level paths for each prefix [36]. These statistics all suggest that large ISPs often have many downstream routes to choose from.

B. *Different Paths Have Different Properties*

The many alternative 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, can cause serious, even disastrous, damage (e.g., in case of online banking) to network users [21]. It was 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 seen by a large ISP [41, 21].

Performance: Path performance (e.g., delay, 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—AS-path length—is a poor indicator of path performance [29, 31, 32]. As a result, alternative BGP paths often have significantly better performance than the default paths [11]. Large ISPs can select better performing paths by leveraging their path diversity [11]. Although some intelligent route control products exist for multi-homed enterprise networks [9], there is no similar counterpart solution in large carrier ISPs.

C. *Path Diversity Calls for 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 route that is "best" in every respect. Therefore, the ISP must synthesize the importance of each objective in specifying an overall policy for selecting the best route.

However, the current BGP decision process imposes inherent restrictions on the policies an ISP can realize [26]. 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 *trade-offs* between policy objectives. For example, a useful policy that strikes a balance between revenue and route stability could be: "*If all routes are 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 be realized today. 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.

D. *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 hosting interactive applications like online gaming and voice over IP may prefer 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 whatever 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 the same best route for all customers connected at the same edge router, precluding the "win-win" opportunity for large ISPs and their customers.

E. *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 [16], our improvements in flexibility must be made judiciously. Fortunately, an ISP can safely advertise *any* route to customers that are "stub" ASes (i.e., ASes only appear at the end of AS paths) [42]. Since stub ASes do not provide transit service, they do not export the routes they learn to other ASes. Our analysis in [40] 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. 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 set of servers that select BGP routes on behalf of the routers [13, 7, 35, 36]. 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.

III. Routing Architecture

In this section, we present the intra-AS routing architecture of Morpheus, which enables the clean-slate design of the flexible route selection process (Section IV). We propose three changes to the way routes are disseminated and assigned, and the way traffic is forwarded within an AS, which provides the ultimate flexibility to the "inputs" and "outputs" of the route selection problem formulated in the Introduction. These changes enable Morpheus to: (1) have complete visibility of all alternative routes, (2) assign customized routes to different edge routers in the AS and neighboring domains, and (3) assign routes independently of each other without causing forwarding loops. As a result, the route selection process can assign any available route to any ingress link (i.e., neighbor) independently. All three architectural features are incrementally deployable through configuration changes and do not require hardware or software upgrades to existing routers.

A. Complete Visibility of BGP Routes

As discussed in Section II, path diversity is the basis of policy flexibility. However, much of the path diversity of a large ISP remains unused as routers do not have complete visibility of BGP routes [34]. An edge router may learn multiple routes for the same destination prefix through external BGP (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, R3 and R4 each learns two routes to destination D, but can only propagate one to R5 (say, the one via R6 and R8, respectively). R5, in turn, propagates only one route (say, the one via R8) to R1 and R2. Then, R2 does not learn, and hence cannot use any of the other available routes (via R6, R7, or R9), even if it would have been preferred by the customer C3 (e.g., to avoid having its traffic go through AS B). Such loss of visibility gets even more pronounced in large networks due to the use of route reflectors [34]. Although propagating only one route helps limit control-plane overhead, it imposes significant limitations on flexibility.

Design Decision 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, as shown in Figure 2. Morpheus can obtain full visibility of all available BGP routes through (multi-hop) eBGP sessions with the routers in neighboring ASes, as in the Routing Control Platform [13, 36].[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. For example, in Figure 1, if Morpheus assigns C3 the route through R6 to reach D, it must also propagate the same

[1]Alternatively full visibility of the routes can be obtained through BGP Monitoring Protocol (BMP) sessions [30] with the AS's own edge routers, which is more scalable.

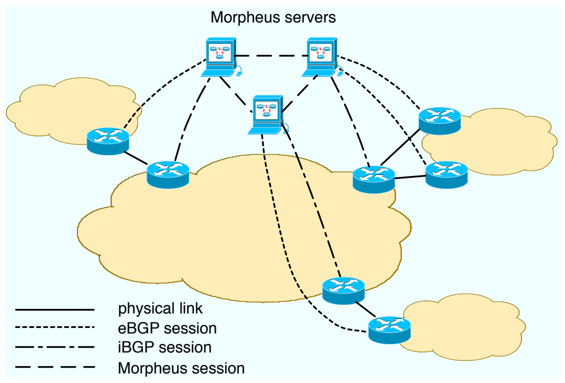


Fig. 2. Morpheus routing architecture: Morpheus servers peer with neighboring domains via multi-hop BGP sessions; edge routers direct interdomain traffic through tunnels.

route to R2 (the edge router C3 is connected to), so that R2 knows how to forward C3's traffic to D using the expected path. Since this architecture uses the BGP protocol itself to learn and assign routes, it does not require any upgrade to the routers in the ISP.

B. Flexible Route Assignment

Even *with* complete visibility of alternative routes, today's BGP-speaking routers cannot assign different paths to different customers. 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 must have the ability to (1) use available paths through any *egress link* (rather than *egress router*) flexibly, and (2) assign those routes to the ingress links *independently* (whether or not they connect to the same edge router).

Design Decision 2: *An AS should be able to assign any route through any egress link to any ingress link 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 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 forwarding-table entries for different customers [25].

[2]This can be achieved by using the "route target" attributes commonly used with VRF in MPLS-VPN [25], or having multiple iBGP sessions between a Morpheus server and an edge router. Other options include using the BGP "add-paths" capability [38].

C. Consistent Packet Forwarding

With the flexibility of assigning any route through any egress link to any neighbor independently, extra care needs be taken in the data plane to avoid introducing forwarding loops. When a router has multiple "equally good" routes, it is common practice to pick the route through the "closest" egress point, based on the Interior Gateway Protocol (IGP) weights, a.k.a. hot-potato routing. For example, in Figure 1, if the routes to D through link R3-R6 and link R4-R9 have the same local preference and AS-path length, and if R1 is closer to R3 than to R4 (in terms of IGP weights), R1 will pick the route through R3-R6. Hot-potato routing ensures consistent forwarding decisions among the routers in the network. For example, if R1 picks the route through R3-R6 to reach D, other routers on the forwarding path (i.e., R5 and R3) are guaranteed to make the same decision.

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 [33].

Design Decision 3: *The routers in an AS should forward packets from the ingress link to its assigned egress link.*

To achieve this goal, Morpheus relies on IP-in-IP or MPLS tunnels to direct traffic between edge links. This design choice offers several important advantages, beyond allowing flexible route assignment without the risk of forwarding anomalies. First, 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. Second, Morpheus does not need to select BGP routes for the internal routers, reducing the total number of routers it has to manage. MPLS or IP-in-IP tunneling is readily available at line rate in many commercial routers, and a "BGP-free core" is increasingly common in large ISPs. In Morpheus, packets are tunneled between edge *links* (rather than between edge routers as is common today). To avoid routers in neighboring domains (e.g., R6 in Figure 1) having to decapsulate packets, edge routers (e.g., R3) need to remove the encapsulation header as part of forwarding the packets, using technique similar to penultimate hop popping [10].

IV. Server Software Architecture

The Morpheus server needs to solve the **route selection problem** introduced in Section I: *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 $\mathcal{C} = \{c_1, c_2, ..., c_k\}$ for each neighboring router.* This problem naturally devolves into two main steps: (i) *classifying* the routes based on each criterion and (ii) *selecting* the best route based on the set of criteria, as shown in Figure 3. Each *policy classifier* tags every received route based on a single policy objective. Each *decision process* picks a best route according to the tags using a "decision function" $\mathcal{F}_\mathcal{C}$ that is configured to realize a particular routing policy. A Morpheus server can run multiple decision processes in parallel, each with a different routing policy, to pick customized routes for different neighbors.

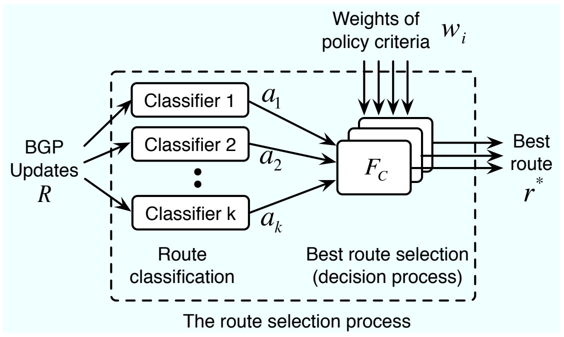


Fig. 3. Morpheus' BGP route selection process, which includes route classification and best route selection.

A. Multiple Independent Policy Classifiers

The introduction of policy classifiers provides flexibility by providing a separate attribute for each policy objective, and incorporating "side information" into route selection.

1) Separate Attribute for Each Policy Objective: The BGP decision process selects best routes by examining one BGP attribute at a time, e.g., first "local-preference", followed by "AS-path length" and so on. As BGP policies involve more and more policy objectives, many of them are forced to be realized by using the same BGP attribute. For example, to realize the common business relationship policy of "prefer customer routes over peer routes, and prefer peer routes over provider routes", customer / peer / provider routes could be assigned with local-preference value of 100 / 90 / 80, respectively. At the same time, operators often increase or decrease the local-preference of a route to make it more or less favorable in the decision process to control the traffic load of certain links. In fact, many other complicated rules are also overloaded to "local preference" via mechanisms such as "route-maps" to *indirectly* influence BGP's multi-stage decision process. The lack of separate attributes for individual policy objectives causes policy configuration to become immensely convoluted, as the attribute overload becomes more severe.

Design Decision 4: *A Morpheus server should use a separate attribute for each policy objective.*

Morpheus' policy classifiers realize this design decision by *tagging the routes*. Each classifier takes 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 latency classifier may tag a route with the measured latency of its forwarding path; a loss classifier may tag a route with the measured loss rate of the path; a stability classifier may tag a route with a penalty score that denotes the instability of the route (using, for example, a route-flap damping algorithm [37]); a security classifier that detects suspicious routes (e.g., those being hijacked) may tag a route as "suspicious" or "unsuspicious" [21].

Each policy classifier works independently and has its own tag space, obviating the need to overload the same attribute. It also makes it 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), sometimes leaving the decision process with no route to choose from.

2) Incorporate Side Information: Another issue that limits the flexibility of routing policies is the lack of *side information*. Many useful routing policies require additional information that is not part of the BGP updates. For example, to select the route with the shortest latency to a destination, we need performance measurement data. (As mentioned in Section II, the AS-path length is a poor indicator of path latency.) In general, side information about route properties includes *external information* such the business relationships with the neighbors, measurement data, or a registry of prefix ownership, and *internal states* such as a history of ASes that originated a prefix (which can be used to detect prefix hijacking [21]), or statistics of route instability. However, there is no systematic mechanism to incorporate side information in routers today. Network operators have to either "hack" their BGP configurations in an indirect and clumsy way (e.g., tweaking "route-maps"), 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.

Design Decision 5: *A Morpheus server should be able to use external information and / or keep internal state when determining the properties of routes.*

The introduction of policy classifiers makes it easy to incorporate side information as each policy classifier can have access to different 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 latency classifier and a loss classifier can get measurement information about path quality from a separate performance monitoring system, or a reputation system (e.g., AS X is well known to have long latency or a high loss rate). A security classifier can have access to a registry of prefixes and their corresponding owners.

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) [21].

Care needs to be taken when taking performance metrics (e.g., latency and loss) into the decision process, as these properties of a path could potentially change quickly with time. Recent studies suggest that it is possible to factor performance into route selection in a stable way [19, 18]. We plan to further investigate the trade-off between route stability and the responsiveness of route selection to performance changes in the context of Morpheus (e.g., use a timer in the classifiers to control how often the performance properties of routes change in the decision process).

B. Multiple Weighted-Sum Decision Processes

The Morpheus server uses a weighted-sum decision process to realize trade-offs amongst different objectives. It also supports running multiple decision processes in parallel to realize different customized policies simultaneously.

1) Weighted-sum for Flexible Trade-offs: The conventional step-by-step BGP decision process imposes a strict ranking of route attributes, starting with local preference and followed by AS-path length and so on. As a result, policies that strike a trade-off among policy objectives are hard to realize, such as the example mentioned in Section I that balances stability and business relationships.

Design Decision 6: *The Morpheus decision process should support trade-offs among policy objectives.*

To achieve this goal, the decision function $\mathcal{F}_{\mathcal{C}}$ in the route selection problem formulation (as mentioned in Section I) must allow trade-offs among policy objectives. A simple, yet powerful method is the *weighted-sum*. For example, for a route $r \in \mathcal{R}$ (where $\mathcal{R}$ is the set of alternative routes), its weighted-sum *score* is:

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

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

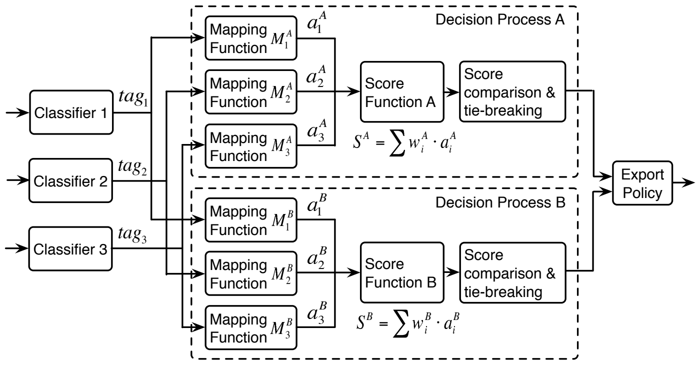


Fig. 4. Each decision process consists of a set of mapping functions of the policy objectives and a score function. Different decision processes are configured with different mapping functions and/or score functions to realize different policies.

$$r^* = \mathcal{F}_\mathcal{C}(r) = \underset{r \in \mathcal{R}(p)}{\arg\max}\, S(r) \qquad (2)$$

We choose the weighted sum as the basis of Morpheus' decision process for three reasons. First, the weighted sum provides an expressive way to make trade-offs between the criteria through the configuration of their weights, and it can also be used to express a sequential process like the standard BGP decision process. Second, weighted sums are simple to compute and thus well-suited to making routing decisions in real time. Third, it allows us to leverage Analytic Hierarchy Process (AHP), a technique in decision theory, to design a simple and intuitive configuration interface, which can automatically derive the weights according to operator's preferences on policy objectives (as discussed in Section V).

Morpheus instantiates one decision process for each routing policy and supports running multiple decision processes in parallel. To allow different decision processes to *interpret* a policy tag differently, each decision process has a set of "mapping functions" before the "score function", as shown in Figure 4. The introduction of the mapping functions offers 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). This way, each policy classifier can tag routes in its own tag space without worrying about the consistency with other classifiers. This facilitates the implementation of classifiers by third parties. With the mapping functions, network operators can simply "plug and play" different classifier modules. The mapping functions can ensure that all tags are converted to the same uniform numerical space to make the comparison between different policy criteria meaningful. We believe this open platform will foster the sharing of classifier modules in the operations community and may also lead in the long run to the emergence of a market centered around these modules.

Second, the mapping functions enables different policies to interpret the same policy tag *differently*. For example, 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 shown in Figure 4, the same tag tag_1 can be mapped to different ratings 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 realize different policies through different configurations of the mapping functions (as well as weights of the policy objectives), as illustrated by the examples in Section V.

After passing the mapping functions, the route is sent to the score function which computes its score, as shown in Figure 4. Then the scores of all the routes for the same destination prefix are compared, and the route with the highest score is picked as the best route. If there are multiple routes with the same highest score, the operators have the choice to break the tie using different mechanisms, such as configuring a (potentially different) ranking of egress links for each ingress link, and pick the route with the highest egress link ranking as the best route [40]; or simply using router ID.

2) Parallel Decision Processes for Customized Policies: BGP allows an AS to influence how other ASes reach itself (e.g., through the use of BGP communities). However, BGP provides no mechanism for an AS to influence how its provider picks routes for it to reach the rest of the Internet. However, such coordination is increasingly important as more customers want routes with particular properties (e.g., low latency, high bandwidth, good security). For example, many content providers (e.g., social network Web sites) rely on their ISPs to reach their users (i.e., the "eyeballs"). To get closer to the "eyeballs", content providers commonly buy services from multiple transit providers and use only the routes that meet their performance requirements. This is not economical for the content provider. A transit provider that could flexibly assign the routes based on customers' preferences would have an advantage over other ISPs in attracting customers.

Design Decision 7: *An AS should allow its neighbors (e.g., its customers) to influence its routing policies by specifying their preferences.*

To support different customer choices, Morpheus supports the realization of multiple independent routing policies simultaneously, through the parallel execution of multiple decision processes, each selecting its own best routes, as shown in Figure 4.

To avoid changing the BGP protocol, Morpheus uses an out-of-band communication channel for customers to specify preferences through a simple configuration interface. For example, the provider could allow a customer to independently and directly configure the weights in a decision process. Alternatively, the provider could combine the customers' preferences between certain policy objectives, and combine them with its own preferences through an AHP-based configuration interface (as discussed in Section V). While providing a separate decision process for each customer may introduce scalability challenges, we believe in practice, the routes most customers want can be reduced to a handful of types, such as low-latency routes, most secure routes, most stable routes, low-cost routes. The provider could simply provide these options to its customers, and only provide customized decision processes to a very limited number of customers who demand more control of their routes.

In any case, Morpheus provides an AS the ability to select routes based on a variety of factors. However, this extra flexibility should not come at the expense of global routing instability. Instability could arise if a collection of ASes have conflicing routing policies or repeatedly adjust their policies in response to performance changes. To prevent policy conflicts, an ISP can adhere to the policy guidelines outlined in [15, 39] and limit more flexible route assignment to stub ASes, as discussed in Section II-E. In our ongoing work, we are investigating how much extra flexibility an ISP can safely provide its other neighbors. To prevent oscillations in interdomain load-sensitive routing, we are exploring possible extensions of recent stable load-balancing techniques [12, 20, 14, 18, 4]. In both cases, considerable flexibility in interdomain routing is possible without compromising global stability.

V. AHP-Based Policy Configurations

In this section, we present how to configure routing policies in Morpheus. In theory, operators could configure the mapping functions and the weights directly to realize policies. However, humans are not good at setting a large number of weights directly to reflect their preferences. Instead, studies show that humans do a much better job in expressing their preferences through pairwise comparisons between alternatives, even though the results of these comparisons are often inconsistent [28]. Based on this observation, Morpheus leverages the Analytic Hierarchy Process (AHP) [28], a technique in decision theory, to provide a simple, intuitive configuration interface. Network operators specify their policy preferences through pair-wise comparisons, and AHP automatically derives the weights of policy objectives and the appropriate ratings of the mapping functions. After briefly explaining how AHP works in an "offline" fashion, we propose an "online" version that is more appropriate for real-time route selection. We then show a policy configuration example, in which the ISP allows its customer to configure part of the decision process. At the same time, the ISP itself controls how much influence on the decision process the customer can have.

A. The Offline AHP Configuration Process

AHP is a well-studied, widely-applied technique in Multi-Criteria Decision Analysis [5], a field in decision theory. It provides a simple, yet systematic way to find the overall best choice from all alternatives, according to the decision maker's preferences of the alternatives with regard to individual criteria [28]. In interdomain routing policy, the alternatives are the available routes, the decision maker is the network operator, and the criteria are the policy objectives.

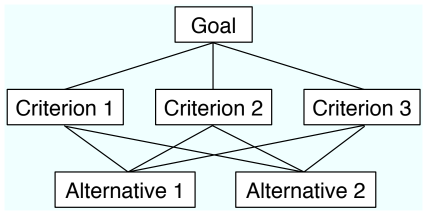


Fig. 5. The decision hierarchy of AHP.

The first step in AHP is to model the decision problem as a *decision hierarchy*, as shown in Figure 5. At the bottom of the hierarchy are the *alternatives*, i.e., the possible solutions of the decision problem. One solution must be selected among the alternatives based on a set of *criteria*, as shown in the middle of the hierarchy. For each criterion, the decision maker then performs pair-wise comparisons of all alternatives. For each comparison, the decision maker specifies his/her preference of one alternative over the other using a number. The scale from 1 to 9 has proven to be the most appropriate [28], in which, 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 extreme preference. 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.

TABLE I
Comparison matrix

Loss Rate	$R1$ (0.01)	$R2$ (0.03)	$R3$ (0.05)	Weight
$R1$ (0.01)	1	3	9	0.69
$R2$ (0.03)	1/3	1	3	0.23
$R3$ (0.05)	1/9	1/3	1	0.08

An example is shown in Table I, where three alternative routes $R1$, $R2$, and $R3$ are compared in pairs based on their loss rate. Note that although the table shows the entire matrix of 9 preferences, the operator only needs to specify 3 of them—"$R1$ vs. $R2$", "$R1$ vs. $R3$", and "$R2$ vs. $R3$". Here the operator weakly prefers $R1$ (with a loss rate of 0.01) over $R2$ (with a loss rate of 0.03); strongly prefers $R1$ over $R3$ (with a loss rate of 0.05); and weakly prefers $R2$ over $R3$. The table also shows the weights of all alternatives, which are computed from the principal eigenvector of the preference matrix [28]. In this case, the operator's preferences are "consistent", i.e., "$R1$ vs. $R3$ (9)" = "$R1$ vs. $R2$ (3)" $\times$ "$R2$ vs. $R3$ (3)", so the weights can be derived by normalizing the values in any column of the preference matrix. However, humans are likely to give *inconsistent* answers in a series of pair-wise comparisons, and AHP provides a systematic way to deal with inconsistency, as illustrated in the example in Section V-C.

With operator's preference of alternative routes on each criterion (e.g., business relationships, latency and loss rate in

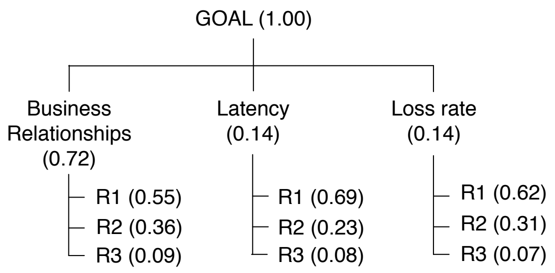


Fig. 6. Example of a decision hierarchy.

Figure 6), AHP can derive the rating $a_i(r)$ of route r for each criterion i, as in Equation (1). To get the weight w_i of each criterion i, the operator also needs to determine the preference (relative importance) of different criteria through similar pair-wise comparisons of criteria. With the preferences of all criteria pairs, AHP can derive the appropriate weight for every criterion, and calculate the overall score of an alternative route using Equation (1). For example, in the hierarchy shown in Figure 6, $S(R1) = 0.72 \times 0.55 + 0.14 \times 0.69 + 0.14 \times 0.62 = 0.58$.

B. Adapting AHP to Work Online

Applying the conventional AHP technique to the route selection problem directly, as described in Section V-A, only works in an *offline* fashion. This is because whenever a new route is received, a human operator has to compare all alternatives routes in pairs with regard to every policy objective (to get the rating $a_i(r)$), which can not be done in real time.

To make the AHP-based decision process work *online*, we replace the alternatives in the decision hierarchy with a set of *subcriteria*. For example, in Figure 7, the business relationships criterion can be divided into three subcriteria: customer, peer, and provider. This change allows network operators to specify their preferences on each set of subcriteria offline, while enabling the ratings $a_i(r)$ of received routes to be generated in real time. For example, for the business-relationship criterion, an operator can specify his/her preference of customer/peer/provider routes through pair-wise comparisons offline. The appropriate rating for each type of route will be derived by AHP automatically and stored in the mapping function (as shown in Figure 4).

In summary, the online, AHP-based policy configuration process can be performed in three steps: (1) **Decompose:** The network operator formulates the decision problem by identifying a hierarchy of criteria (and subcriteria); (2) **Specify preferences:** For each pair of criteria at the same level of the hierarchy and with the same "parent criterion", the network operator specifies his/her preference of one criterion over the other; (3) **Derive weights:** The preferences are organized in preference matrices and weights are derived by AHP using linear algebra operations [28]. Note that operators are only involved in the first two steps, and the third step is performed by the configuration program automatically.

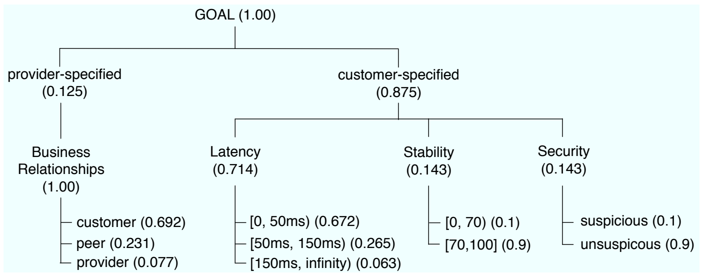


Fig. 7. The AHP hierarchy of an example routing policy.

C. A Policy Configuration Example

As mentioned in Section IV, Morpheus enables an ISP to get input from its customers about their preferences on routes. Here we give an example that shows how customer preference can be incorporated into the decision process using the AHP-based configuration interface.

Suppose the ISP has a customer C who is a content provider, and C has purchased the "premium service" that allows it to specify its preference on the routes it learns from the ISP. As a content provider, C is primarily interested in learning routes that have low latency to the destinations (i.e., to get the content closer to the "eyeballs"). The ISP, on the other hand, cares about the "business relationships" property of the routes, as it would earn profit by forwarding traffic through a customer, and it would have to pay to forward traffic through a provider.

Figure 7 shows the AHP hierarchy of the routing policy, which takes four policy objectives into account: business relationships, latency, stability, and security. As the first step of the configuration, the ISP needs to decide how much influence to the decision process it gives to customer C. As a premium service, the ISP allows C to directly specify its preferences on all policy objectives except business relationships. It also strongly prefers the customer-specified objectives over the provider-specified objective, and enters "7" in the "customer-specified vs. provider-specified" comparison. AHP then automatically derives the relative weights of the two types of objectives: 0.875 for the three customer-specified objectives (latency, stability, and security) and 0.125 for the provider-specified objective (business relationships).

To determine the relative weights of latency, stability, and security, the customer C needs to specify its preferences through pair-wise comparisons. Assuming that C enters "latency vs. stability" = 5, "performance vs. security" = 5, and "stability vs. security" = 1, AHP can then derive the weights of the three objectives: latency (0.714), stability (0.143), and security (0.143), as shown in Figure 7.

Now that the weights of the four policy objectives are derived, the ISP and the customer C only need to configure the corresponding mapping functions for the objectives. Assuming that the ISP specifies its preferences on business relationships as: "customer vs. peer" = 3, "peer vs. provider" = 3, and "customer vs. provider" = 9, then AHP automatically derives the ratings of the three types of routes for the mapping function of business relationships. Upon receiving a route tagged as “customer”, “peer”, or “provider” by the business relationship classifier, the mapping function will assign it with a business relationship rating of 0.692, 0.231, or 0.077, respectively.

For the latency mapping function, suppose the customer C is given three latency intervals: i_1 = $[0, 50msec]$, i_2 = $[50msec, 150msec]$, and i_3 = $[150msec, \infty]$, and it has the following preferences: “i_1 vs. i_2” = 5, “i_1 vs. i_3” = 9, and “i_2 vs. i_3” = 3. AHP will then derive the ratings the mapping function should use to map the routes that fall into the three intervals: i_1 = 0.672, i_2 = 0.265, and i_3 = 0.063. While calculating the ratings, AHP also calculates the *consistency ratio* of the preferences [28], where a consistency ratio of 0 means all preferences are consistent. In this case, the three preferences are inconsistent (i.e., “i_1 vs. i_3” $\neq$ “i_1 vs. i_2” $\times$ “i_2 vs. i_3”), and the consistency ratio is 0.028. AHP requires the consistency ratio to be no larger than 0.05 (n = 3), 0.08 (n = 4), or 0.1 ($n \geq 5$) for a set of preferences to be acceptable, where n is the number of alternatives [28]. (As 0.028 is below the 0.05 threshold, this set of preferences is acceptable.) When a set of preferences specified by an operator has a consistency ratio larger than the threshold, Morpheus will request the operator to reset the preferences.

For stability, we assume the stability classifier runs an algorithm similar to the one used by route-flap damping (RFD), and tags each route with a number between 0 and 100. The higher the number is, the more stable the route is. The customer C treats routes with a stability tag below 70 as unstable, and it extremely prefers stable routes over unstable ones. For security, we assume the security classifier runs the Pretty-Good BGP (PG-BGP) [21] algorithm, and tags every route as either “suspicious” or “unsuspicious”. The customer C extremely prefers unsuspicious routes over suspicious routes.

In a similar fashion, the provider can provide customized routing policies to different customers using separate decision processes (as shown in Figure 4), and allow each customer to configure certain policy objectives through the simple AHP-based interface.

VI. Implementation and Evaluation

We have implemented a Morpheus prototype as an extension to the XORP software router platform [17]. In this section, we first highlight the major changes we made to XORP, then evaluate the performance and scalability of Morpheus using our XORP-based prototype. Specifically, we answer three questions:

1. *What is the performance of Morpheus’ policy classifiers and its score-based decision process?* 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 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 [36].

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.

Due to space limitation, here we only present the evaluation details of the first question, and leave the details of the second and third questions to [40].

A. Prototype as an Extension to XORP

We chose XORP as the platform to implement our Morpheus prototype because its modular structure closely matches the Morpheus software architecture. However, since XORP is designed to implement the standard BGP decision process for individual routers, our prototype differs from XORP’s 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 and peers, and can simultaneously run multiple decision processes each having its own policy configuration.

Second, to demonstrate that policy classifiers are easy to implement and to evaluate their performance, we implemented four policy classifiers for business relationships, latency, stability, and security, respectively. While these classifiers could, in principle, work in parallel, we implemented them as new modules in the XORP message-processing pipeline. Since the classifiers work independently, the ordering amongst them is not important.

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.

B. Evaluation 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 [27] 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.

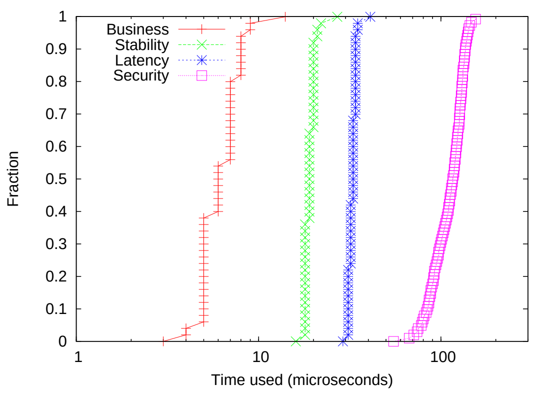


Fig. 8. Classification time: time taken by the classifiers to tag a route.

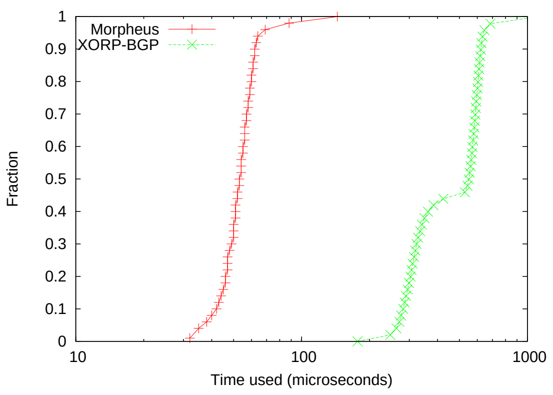


Fig. 9. Decision time: comparison between Morpheus and XORP-BGP, 20 routes per prefix.

C. Evaluation of 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 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 with a rank-based tie-breaking step [3] (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 classifier is fed with static tables of path latency data. We believe the result 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 8, 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 PG-BGP 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 (excluding the classification 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.

[3]In the rank-based tie-breaking scheme, each edge router is assigned with a fixed (but configurable) ranking of all egress points, and the edge router with the highest ranking is selected as the winner [40].

Decision time (multiple alternative routes per prefix): In the next experiment, we compare the decision time of Morpheus and the out-of-the-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 a fair comparison, we configure Morpheus to use router ID to break ties. From Figure 9 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 smaller variation, while XORP-BGP's decision time varies significantly, ranging from less than 100 microseconds (when there is only a small number of alternative routes for a prefix) to over 500 microseconds (when the number becomes large).

TABLE II
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 (i.e., $\binom{20}{2} \cdot 0.01^2 = 0.019$) 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 II. 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.

VII. Related Work

Previous work proposes to raise the level of abstraction of BGP policy configuration through network-wide, vendor-neutral specification languages [2, 6]. 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 of "design for configurability" and present a system that supports more flexible routing policies and yet is easier to configure.

Several recent studies on the Routing Control Platform (RCP) [13] advocate moving the BGP control plane of a single AS to a small set of servers that select routes on behalf of the routers [7, 35, 36, 3]. The prototype systems in [7] and [36] 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, the system in [7] simply mimics the standard BGP decision process, without expanding the space of realizable policies. While [35] and [36] support more flexible alternatives to today's hot-potato routing, these systems do not create an extensible framework for realizing flexible policies with trade-offs amongst policy objectives, or support multiple different policies simultaneously. They do not revisit the convoluted BGP configuration interface either. These are the main contributions of our Morpheus design.

VIII. 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. 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. It also enables an ISP to allow its customers to influence its routing policies through a simple and intuitive configuration interface. 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 ratings 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, although some customers will subscribe to customized routes, the remaining customers will still 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, 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.

Acknowledgment

The authors gratefully acknowledge the support of HSARPA/ARO (Grant W911NF-05-1-0417) and NSF (Grant CNS-*0519885). The authors would also like to thank the anonymous reviewers for their comments. Ioannis Avramopoulos was with Princeton University while performing this work.

References

[1] http://www.networksorcery.com/enp/protocol/bgp.htm.

[2] 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.

[3] H. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. V. der Merwe. Anycast cdns revisited. In *Proc. International World Wide Web Conference*, 2008.

[4] I. Avramopoulos, J. Rexford, and R. Schapire. From optimization to regret minimization and back again. In *Proc. Third Workshop on Tackling Computer System Problems with Machine Learning Techniques (SysML08)*, December 2008.

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

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

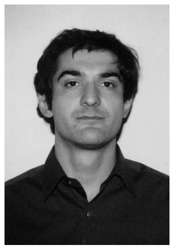
[7] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a Routing Control Platform. In *Proc. Networked Systems Design and Implementation*, May 2005.

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

[9] Cisco Optimized Edge Routing. http://www.cisco.com/en/US/products/ps6628/products_ios_protocol_option_home.html.

[10] MPLS Fundamentals: Forwarding Labeled Packets. http://www.ciscopress.com/articles/article.asp?p=680824&seqNum=2.
[11] 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.
[12] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*, 2001.
[13] 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.
[14] S. Fischer, N. Kammenhuber, and A. Feldmann. REPLEX — dynamic traffic engineering based on wardrop routing policies. In *Proc. CoNext*, December 2006.
[15] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Trans. Networking*, December 2001.
[16] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Networking*, 10(1):232–243, 2002.
[17] 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.
[18] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang. Rethinking Internet traffic management: From multiple decompositions to a practical protocol. In *Proc. CoNext*, December 2007.
[19] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, 2005.
[20] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *Proc. ACM SIGCOMM*, August 2005.
[21] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks, Special issue on Complex Computer and Communications Networks*, October 2008.
[22] R. Mahajan, D. Wetherall, and T. Anderson. Mutually controlled routing with independent ISPs. In *Proc. Networked Systems Design and Implementation*, 2007.
[23] 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.
[24] Discussion on NANOG mailing list. http://www.merit.edu/mail.archives/nanog/2007-04/msg00502.html.
[25] I. Pepelnjak and J. Guichard. *MPLS and VPN Architectures*. Cisco Press, 2000.
[26] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC 4271, January 2006.
[27] The routeviews project. www.routeviews.org.
[28] T. L. Saaty. *The Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process, Vol. VI, AHP Series*. RWS Publications, 2000.
[29] 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.
[30] J. Scudder. BGP monitoring protocol. Expired Internet Draft draft-scudder-bmp-00, Augeust 2005.
[31] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proc. ACM SIGCOMM*, 2003.
[32] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proc. SPIE ITCom*, 2001.
[33] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, June 2004.
[34] 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.
[35] 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.
[36] 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.
[37] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping. RFC 2439, November 1998.
[38] D. Walton, A. Retana, and E. Chen. Advertisement of multiple paths in BGP. Expired Internet Draft draft-walton-bgp-add-paths-05, March 2006.
[39] H. Wang, H. Xie, Y. R. Yang, L. E. Li, Y. Liu, and A. Silberschatz. On stable route selection for interdomain traffic engineering: Models and analysis. Technical Report YALEU/DCS/TR-1316, Dept. of Computer Science, Yale Univ., February 2005.
[40] Y. Wang, I. Avramopoulos, and J. Rexford. Morpheus: Enabling flexible interdomain routing policies. Technical Report TR-802-07, Princeton University, October 2007.
[41] 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.
[42] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *Proc. ACM SIGCOMM*, September 2006.

Yi Wang is a 4th-year Ph.D candidate in Department of Computer Science at Princeton University. His research interest is in Internet routing and network management, especially in reducing management complexity of routing for Internet service providers. He has interned at Cisco, AT&T Labs–Research and Microsoft Research Asia. He received his BSE and MSE degrees in electrical engineering from Tsinghua University, Beijing, China, in 2003 and 2005, respectively.



Ioannis Avramopoulos (S'99, M'09) is a Senior Research Scientist at Deutsche Telekom Laboratories in Berlin, Germany. Previously he was a postdoctoral researcher in the Computer Science Department at Princeton University in Princeton, New Jersey. He received a diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1999 and the Master and Ph.D. degrees in Electrical Engineering from Princeton University in 2003 and 2006, respectively. His research interest is in networking and security.

Jennifer Rexford (S'89, M'96, SM'01) is a Professor in the Computer Science department at Princeton University. From 1996-2004, she was a member of the Network Management and Performance department at AT&T Labs–Research. She received her BSE degree in electrical engineering from Princeton University in 1991, and her MSE and PhD degrees in computer science and electrical engineering from the University of Michigan in 1993 and 1996, respectively.