

Wresting Control from BGP: Scalable Fine-grained Route Control

Patrick Verkaik*, Dan Pei†, Tom Scholl†, Aman Shaikh†,
Alex C. Snoeren*, and Jacobus E. van der Merwe†

**University of California, San Diego* †*AT&T Labs – Research*

Abstract

Today’s Internet users and applications are placing increased demands on Internet service providers (ISPs) to deliver fine-grained, flexible route control. To assist network operators in addressing this challenge, we present the Intelligent Route Service Control Point (IRSCP), a route control architecture that allows a network operator to flexibly control routing between the traffic ingresses and egresses within an ISP’s network, without modifying the ISP’s existing routers. In essence, IRSCP subsumes the control plane of an ISP’s network by replacing the distributed BGP decision process of each router in the network with a more flexible, logically centralized, application-controlled route computation. IRSCP supplements the traditional BGP decision process with an *explicitly ranked decision process* that allows route control applications to provide a per-destination, per-router explicit ranking of traffic egresses. We describe our implementation of IRSCP as well as a straightforward set of correctness requirements that prevents routing anomalies. To illustrate the potential of application-controlled route selection, we use our IRSCP prototype to implement a simple form of dynamic customer-traffic load balancing, and demonstrate through emulation that our implementation is scalable.

1 Introduction

Given the best-effort communication model of the Internet, routing has historically been concerned with connectivity; that is, with finding a loop-free path between endpoints. Deviations from this default behavior normally involved policy changes at fairly slow time-scales to effect business and network management objectives [8]. Within a particular network (or autonomous system), routing was realized by a fixed, fairly simple BGP decision process that tries to ensure consistent decision making between the routers in the network, while respecting the network operator’s policies.

As networked applications and traffic engineering techniques have evolved, however, they place increasingly sophisticated requirements on the routing infras-

tructure. For example, applications such as VoIP and on-line gaming can be very sensitive to the characteristics of the actual chosen data path [9, 10, 27]. A number of studies have shown that non-default Internet paths can provide improved performance characteristics [1, 2, 31], suggesting the potential benefit of making routing aware of network conditions [12]. Additionally, today’s operators often wish to restrict the any-to-any connectivity model of the Internet to deal with DDoS attacks. Finally, in some cases the default BGP decision process is at odds with provider and/or customer goals and may, for example, lead to unbalanced egress links for customers that are multi-homed to a provider [34].

These demands have in common the need for route control that is (i) fine-grained, (ii) informed by external information (such as network conditions), and (iii) applied at time-scales much shorter than manual routing configuration changes (i.e., is “online”) and therefore implemented by means of a *route control application*. Unfortunately, BGP does not provide adequate means for performing online, informed, and fine-grained route control. The tuning parameters BGP does provide are both arcane and indirect. Operators and developers of route control applications are forced to manipulate BGP route attributes in cumbersome, vendor-specific router configuration languages at a low level of abstraction, frequently leading to ineffective or, worse, incorrect decisions [24].

We address these requirements by presenting the design and implementation of a distributed Intelligent Route Service Control Point (IRSCP), which subsumes the routing decision process in a platform separate from the routers in a network. We previously introduced the concept of a centralized Route Control Platform (RCP) that is separate from and backwards compatible with existing routers [13]. In more recent work, we also demonstrated that route selection could be informed by “network intelligence” to enable sophisticated connectivity management applications [34]. To reflect this thinking we have changed the name of our architecture to Intelligent Route Service Control Point (IRSCP). The work presented in this paper builds on this earlier work and offers three important new contributions in the evolutionary path towards a new routing infrastructure:

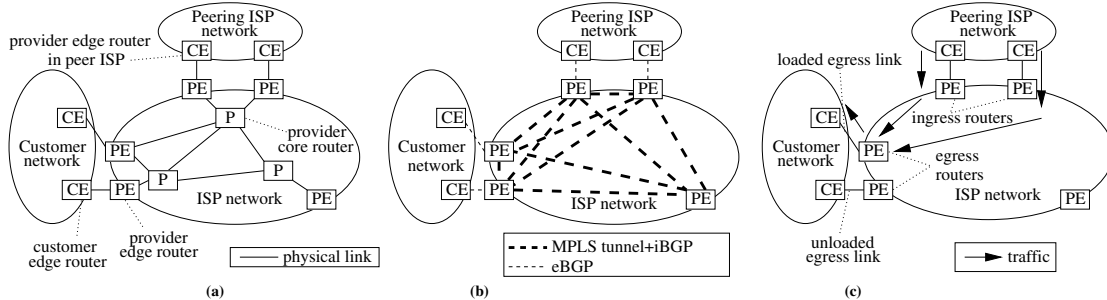


Figure 1: ISP routing infrastructure. (a) Physical connectivity. (b) BGP and MPLS. (c) Traffic ingresses and egresses.

Application-directed route selection: IRSCP eases the realization of applications that dynamically manage connectivity. Specifically, we allow an ISP’s *route control application* to directly control route selection through an intuitive, vendor-independent interface. The interface is based on the abstraction of a *ranking* of egress routes for each router and destination. Note that by “application” we mean network control and management functions that a service provider would employ to provide services. In other words, we are not at this time contemplating third parties providing such IRSCP applications, however, we do envision end users directly interacting with IRSCP applications in a controlled manner to influence the route selection for “their” routes.

Complete route control: IRSCP maintains *complete* control of the route selection function for all routers in the network. As we argue later, this can only be achieved by having IRSCP communicate directly with routers in neighboring networks via eBGP, in addition to speaking iBGP with the routers in the IRSCP-enabled network¹. This gives IRSCP *full visibility* of all routes available in the network. Further, IRSCP is now the *sole* controller of BGP route selection, meaning that all of the network’s routing policy can be handled by the route control application through IRSCP, as opposed to placing some policy configuration on the routers themselves, as is the case in an iBGP-speaking IRSCP [13].

Distributed functionality: Realizing an IRSCP that has complete control of route selection faces two important challenges. First, because routers in the IRSCP-enabled network completely rely on the IRSCP for routing decisions, the IRSCP infrastructure must be substantially more robust than an iBGP-speaking IRSCP. Indeed, a complete failure of an eBGP-speaking IRSCP would essentially cause all routes in the network to be withdrawn, thus effectively incapacitating the underlying network. Second, the need for complete route control and full visibility poses significant scalability challenges. To address these concerns, we partition and distribute the IRSCP

functionality across a number of servers while still ensuring consistent decision making for the platform as a whole. (This is in contrast to simple replication of the same functionality [7], which requires each replica to scale to accommodate the complete network.) Although the IRSCP is physically distributed, it presents a *logically centralized* abstraction from the point of view of a route control application. I.e., while the route control application has to interact with all IRSCP servers, it can remain agnostic to where these servers reside in the network and what set of routers each server controls.

We present a modified BGP decision process, which we call the *explicitly ranked decision process*, together with a route control interface that enables route control applications to directly guide the route selection process in IRSCP. The key challenge to modifying the BGP decision process is to ensure that the resulting protocol retains (or improves) BGP’s robustness, scalability, and consistency properties. We present two simple constraints on the application-provided route ranking that together ensure that IRSCP installs only safe routing configurations, even in the face of router failures or dramatic changes in IGP topology. We see this as a first step towards a “pluggable” route control architecture in which the route control application consists of several independently developed modules and uses the constraints to resolve conflicts, ensuring that only safe routing configurations are sent to IRSCP. We demonstrate the effectiveness of IRSCP’s route control interface by evaluating a sample route control application (consisting of a single module) that uses IRSCP’s interface to load balance customer traffic [34]. Finally, we show through experimentation that our prototype implementation is capable of managing the routing load of a large Tier-1 ISP.

2 Background and motivation

We begin by providing a brief overview of routing and forwarding in a modern MPLS-enabled ISP network. We then motivate the need for application-directed route selection.

¹Following our previous taxonomy [13], the architecture presented in this paper therefore represents a phase-two RCP.

BGP Decision Process (Section 2.1)	Explicitly Ranked DP (Section 2.2)
0. Ignore if egress router unreachable in IGP 1. Highest local preference 2. Lowest AS path length 3. Lowest origin type 4. Lowest MED (with same next-hop AS)	(same)
B-5. eBGP-learned over iBGP-learned B-6. Lowest IGP distance to egress router B-7. Lowest router ID of BGP speaker	R-5. Highest explicit rank R-6. Lowest egress ID

Table 1: Left: the steps of the BGP decision process. Right: the steps of our *explicitly ranked decision process*. Steps 0-4 are identical and produce the *egress set*.

2.1 Current operation

Figure 1(a) shows a simplified view of the physical infrastructure of an MPLS-enabled ISP backbone. The routers at the periphery of the ISP network connect to other ISPs (called peers) and customers. These routers are termed *Provider Edge (PE)* routers, and the routers that interconnect the PE routers are called *Provider Core (P)* routers. The customer routers connecting to PEs are called *Customer Edge (CE)* routers. For simplicity we also use *CE* to represent *peer* routers that connect to the ISP. BGP allows an ISP to learn about destinations reachable through its customers and peers. Typically every PE maintains BGP sessions with its attached CEs, and also with other PEs in the ISP network; the former are known as *eBGP* (external BGP) sessions and the latter as *iBGP* (internal BGP) sessions, as shown in Figure 1(b). When a PE router receives a route through its eBGP session, it propagates the route to other PEs through iBGP sessions, allowing every PE to learn how to reach every peer or customer network. The path traffic follows between *ingress* and *egress* routers is determined by another routing protocol known as an interior gateway protocol (IGP), such as OSPF. In an MPLS network, label-switched paths are established between all PEs (see Figure 1(b)), obviating the need to run BGP on *P* routers. Note that while MPLS is used for setting up and managing tunnels, the actual tunnel path over which a tunnel is set up is determined by the IGP.

A PE usually receives more than one egress route for a given destination and must run a route selection algorithm called the *BGP decision process* to select the best route to use for data forwarding. The BGP decision process (shown in the first column of Table 1) consists of a series of steps. Starting with the set of routes available to the PE, each step compares a set of routes and passes the most preferred routes to the next step while discarding the remaining routes. Steps 1–4 compare routes in terms of *BGP attributes* attached to the routes, while steps 0 and B-6 consider the IGP information associated with the egress PE of the route. We call the set of routes remaining after Steps 0–4 the *egress set*. Steps B-5 and B-6 are responsible for what is called *hot-potato routing*, i.e., for-

warding traffic to the nearest (in terms of IGP distance) egress PE in the egress set. Step B-7 is a tie-breaker that ensures that the PE always ends up with a single best route.

Network operators may configure *policy* statements on PEs to filter undesired routes or modify attributes of routes so as to influence the BGP decision process [8]. Policy can be applied on accepting routes from neighbors (*import policy*) and before sending routes to neighbors (*export policies*). Typically ISPs apply policies only on eBGP sessions and rarely on iBGP sessions.

2.2 Application-directed route selection

We use Figure 1 to highlight a specific problem introduced by BGP’s hot-potato routing thereby motivating the need to enhance route selection with fine-grained application-directed route control.² We assume that the customer shown in Figure 1 has multihomed to the provider network for the purpose of improving redundancy, and so the same destinations are reachable via both links. (This is common practice for data centers and other customer networks with high availability requirements.) All PEs in the provider network therefore have two possible routes to reach the customer network. Assume further that most of the traffic destined to the customer network enters the provider network from the peering ISP network. Assuming unit IGP costs for each internal provider link in Figure 1(a), the two ingress PEs at the top of the figure prefer the route via the top egress PE connected to the customer network (Figure 1(c)). This leads to an imbalance in the load on the two egress links, with the top egress link carrying all (or most) of the traffic, which in turn may result in congestion.

In practice the customer or ISP may prefer the ISP to load-balance traffic on the two egress links while still considering the IGP path cost between ingress and egress. In IRSCP this goal is achieved by basing the decision process on the input from a load-balancing *route control application* run by the ISP, which takes into account IGP path cost, as well as “offered” ingress load and capacity of egress links. The route control application directly controls route selection for each PE and, in this example, directs the two ingress PEs to each send traffic to a different egress PE. We emphasize that this routing solution cannot be achieved in BGP without manipulating BGP attributes on multiple PEs through vendor-specific configuration languages. While it is technically possible to implement the specific load-balancing application we describe using existing BGP mechanisms, the required configuration changes are both complicated and fragile. For example, a system of per-PE policy rules

²A similar problem is introduced by *cold-potato* routing based on the BGP MED attribute (Step 4 in Table 1).

could be devised that manipulate BGP attributes (in particular “local preference”, see Table 1) in such a way that the PE prefers a particular route over other routes. However, since BGP attributes have network-wide scope, the attribute changes must then be reset when the route is advertised to other routers to prevent interference with their decision process. In contrast, IRSCP exports a simple, intuitive interface that allows a route control application to supply a *ranking* of egress links that is evaluated during execution of a modified decision process, the *explicitly ranked decision process*. The interface effectively isolates complexity and intelligence in the route control application, while IRSCP itself remains simple. As shown in Table 1, the explicitly ranked decision process replaces the hot-potato steps (B-5 and B-6). As a result, our modified decision process honors the impact of existing BGP attributes on the decision process as defined by BGP (Steps 1–4).

3 Scalable route control

In this section we describe the design of a distributed Intelligent Route Service Control Point (IRSCP). We begin by presenting our modified route selection process that gives route control applications the ability to direct traffic entering the network at a given ingress PE to an arbitrary egress link. The second section presents the architecture of IRSCP, discussing issues of scalability and fault-tolerance. We formulate a consistency requirement for the explicitly ranked decision process that prevents forwarding anomalies and show that enforcing simple constraints on the application input is sufficient to satisfy the requirement.

3.1 Explicitly ranked decision process

IRSCP implements two types of decision process: the normal BGP decision process and the *explicitly ranked decision process*. Both perform route selection on behalf of individual PEs and so are defined on a per-PE basis. The BGP decision process is used for the subset of destinations, *unranked prefixes*, for which the customer, ISP or route control application has determined that conventional hot-potato routing can be used. For the remaining *ranked prefixes*, the route control application creates a preference ranking of egress routes for each ingress router, the selection of which is realized in IRSCP by the explicitly ranked decision process.

In our architecture the route control application tells IRSCP which routers should use which egress routes based on routing information, traffic load measurements, etc. As a general principle IRSCP follows the directives of the application, except when doing so severely im-

pairs connectivity. An instance of this principle is that we let the application specify a *ranking* of egress links, i.e., egress links ranked by preference, rather than a fixed assignment of egress links to routers. (Each egress route corresponds to only one egress link, and we therefore use the terms egress link and egress route interchangeably.) Using a ranking accommodates unavailability of egress routes. For example, if the top-ranked egress route is unavailable, the next-ranked egress route may be selected. The application specifies the ranking on a per-destination, per-router basis.

We construct our decision process for ranked prefixes (Table 1) by adopting Steps 0–4 of the BGP decision process and then apply the explicit ranking instead of performing hot-potato routing, followed by a tie-breaker in Step R-6.³ This ensures that the explicitly ranked decision process respects BGP attributes such as AS path length (Step 2) and that it takes reachability of egress routers into account. In principle, the explicit ranking can be applied at any point in the decision process, e.g., it may override earlier steps of the decision process. However, we leave exploring the extent to which we may safely override or replace BGP attributes to future work. As we explain in Section 3.2.4, we specifically do not include a step based on IGP distance (Step B-6).

We explore an example of the explicitly ranked decision process by considering the scenarios shown in Figures 2(a) and (b). In this example a single IRSCP server runs the decision process for every PE in the ISP’s network. We examine the execution of the decision process for PE A in Figure 2(a). First the IRSCP server receives all routes for the given prefix: $E - C$, $F - C$ and $G - D$. (We refer to each route using its *egress ID*, the pair of (CE,PE) routers incident on the egress link for the route.) Next, the explicitly ranked decision process for PE A executes Steps 0–4 and in Step 2 eliminates egress route $F - C$ based on the longer AS path length. (We assume that the routes otherwise have identical BGP attributes.) The result is the egress set $\{E - C, G - D\}$. In Step R-5 the decision process applies the explicit ranking for PE A to the egress set. Since the top-ranked egress link $E - C$ is present in the egress set, the decision process selects this route for PE A . Similarly, the decision process selects route $E - C$ for PE C , and route $G - D$ for PEs B and D , resulting in the forwarding behavior shown in Figure 2(b). An important observation is that Steps 0-4 are identical for all PEs. Therefore the decision process for any PE computes the same egress set.

³In practice we also add a tie-breaker on “IRSCP ID” at the end of the explicitly ranked decision process, based upon the IRSCP server that receives the route on an eBGP session.

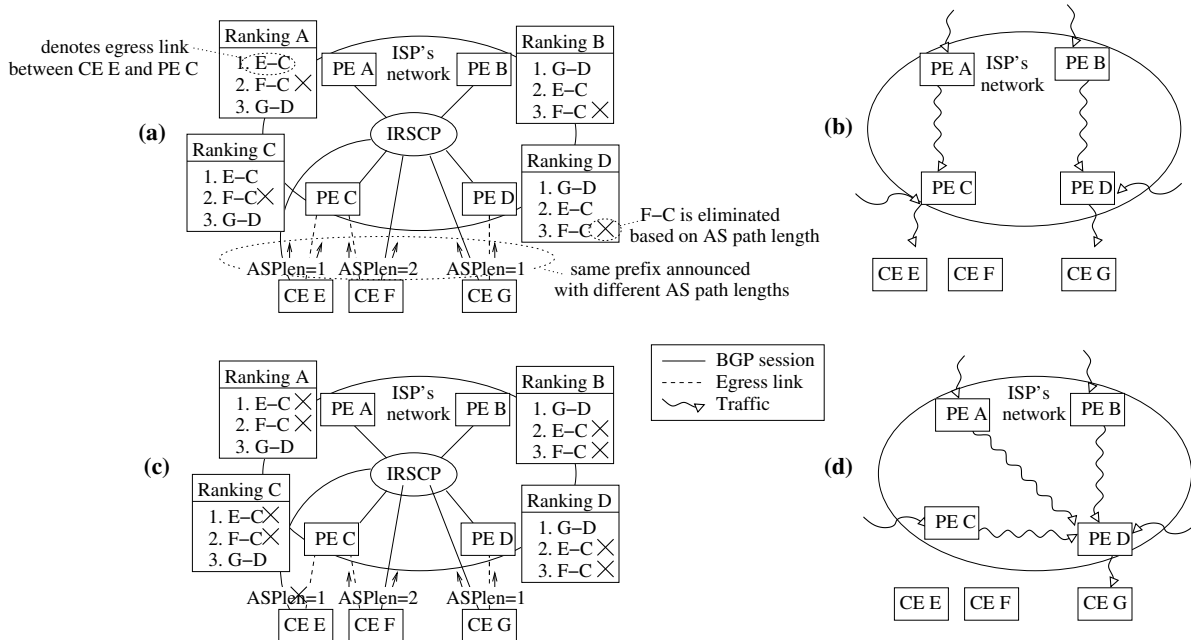


Figure 2: Example of ranking and its effect on forwarding. The application (not shown) has provided IRSCP with the explicit rankings indicated. Each ranking is a list of egress IDs. CEs E , F and G announce routes for the same prefix (with different AS path lengths) through their eBGP sessions with IRSCP. (a) Initial scenario. (b) Forwarding behavior for (a). (c) CE E withdraws its routes. (d) Resulting forwarding behavior.

3.1.1 Outdated rankings

Ideally, the input from the application to IRSCP continuously reflects the current state of the network. In practice, however, IRSCP, being an active participant in BGP, is in a better position to respond instantly to changes in routing state such as BGP route attributes (using Steps 0–4 of the decision process), IGP distances (discussed in Section 3.2.4), and the availability of BGP routes (discussed below). IRSCP must therefore adapt the rankings from the application (based on possibly outdated routing information) to current routing information, as follows.

Between the time that the application sends the rankings to IRSCP and the time that the explicitly ranked decision process runs, new egress routes may be announced and old routes may be withdrawn. Until the application updates its rankings, IRSCP must accommodate discrepancies between the available routes assumed when the application creates the rankings and the actual available routes. An instance of a withdrawn egress route is illustrated in Figures 2(c) and (d), in which CE E withdraws egress route $E - C$, and the egress set changes to $\{G - D\}$. As a result, the decision process changes its selection for PEs A and C to $G - D$ and all traffic egresses through PE D (Figure 2(d)). In other words, a ranking specifies not only the desired routing for the PE in the absence of failure, but also the desired fail-over behavior that the PE should adopt.

Conversely, if new egress routes are advertised, IRSCP

appends them to the end of the explicit ranking (in order of egress ID) until the application is able to provide a revised ranking (Steps R-5 and R-6). Alternatively, the application may *prevent* IRSCP from appending routes in this manner. For example, the application may wish to restrict the set of egress routes of a particular customer to a fixed set, thereby preventing some forms of prefix hijacking. We define a “virtual” *black-hole egress route*, which is part of every egress set and (conceptually) sinks traffic directed to it. We also define a corresponding *black-hole egress ID*, which an application can include as part of a PE’s ranking. If the explicitly ranked decision process for a PE selects the black-hole egress route, the IRSCP server does not send a route to the PE (or its attached CEs), thus making the destination unavailable through that PE. (Step R-6 of the explicitly ranked decision process considers the value of the black-hole egress lower than any other egress ID, effectively ignoring it.)

3.1.2 Other IRSCP applications

IRSCP gives route control applications the ability to direct traffic entering the network at a given ingress PE to an arbitrary egress link. In our current design, IRSCP directs traffic at the granularity of a destination and an ingress PE. However, recently proposed extensions to BGP [25] (together with appropriate data plane support) allow IRSCP to facilitate flow-specific forwarding in addition to destination-based forwarding, e.g., to differen-

tiate VoIP from Web traffic. Similarly, using VPN technology, routing can be controlled at the level of ingress link (i.e., source customer) rather than ingress router, by manipulating an ingress-specific virtual routing and forwarding (VRF) table.

By opening up the decision process to external input, IRSCP enables a class of applications in which routing is controlled based on information external to BGP. An example of an application that uses external information (or in this case analysis) is presented in [22]. The authors propose a pragmatic approach to BGP security by which suspicious routes are quarantined for a certain period of time before being considered for selection. In this case knowledge about the historical availability of routes (even if the routes were not selected) is important to determine whether a route is potentially hijacked. Further, IRSCP provides a mechanism (the black-hole egress route) by which routers can be prevented from selecting suspicious routes until deemed safe by the application.

Another example in this category is the load-balancing application described above, which makes use of network conditions inside the IRSCP-enabled network to inform route selection. However, it is also possible to inform route selection with network conditions external to the IRSCP-enabled network. For example, Duffield *et al.* [12] explore the possibility of using measured path conditions to select between various alternate paths leading to the same destination. This allows the network to route traffic that is sensitive to adverse network conditions along paths more favorable than those that default BGP would be capable of finding (to the extent permitted by the policies encoded in BGP attributes).

The complete route visibility afforded by IRSCP also simplifies a number of route monitoring applications. For example, auditing applications that ensure that peers abide by peering agreements require complete visibility of the routes being advertised to a network [28]. Similarly, “what-if” network analysis [14] becomes much simpler if the application is aware of all the routes that were available. IRSCP’s ability to use external input to inform route selection, however, is its key advantage.

3.2 IRSCP architecture

Figure 3 contrasts logical and physical views of the IRSCP architecture. In Figure 3(a) an application uses “external information” to inform route selection in IRSCP, which in turn communicates selected routes to the routers in the ISP network and in neighboring ISP networks. Figure 3(b) shows a simplified physical realization of the IRSCP architecture, consisting of the route control application and a distributed set of IRSCP servers that collectively perform route selection. The function of IRSCP is to compute a routing solution in which

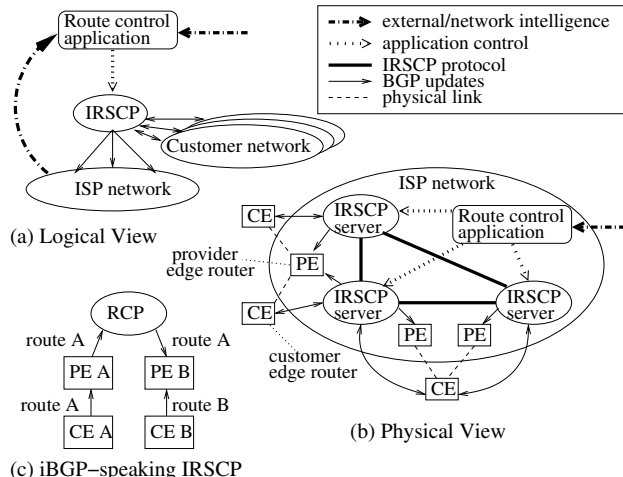


Figure 3: Overview of the IRSCP architecture.

egress routes received from CE routers are assigned to PE routers, so that a PE router will forward traffic it receives along the route. IRSCP performs this function by communicating with PEs and CEs using standard BGP, as a result of which customers and peer networks need not be aware that their CEs peer with IRSCP rather than BGP routers. Specifically, IRSCP receives BGP routes from CE routers over eBGP, executes a per-PE decision process to determine what routes each PE should use, and sends the resulting BGP routes to the PEs over iBGP. IRSCP also sends an update to each CE attached to a PE (again over eBGP) that corresponds to the routing decision that was made for the PE.

We use Figure 3(c) to emphasize the importance of implementing an *eBGP-speaking* IRSCP in order to establish full route control. The figure shows an IRSCP that only speaks iBGP (similar to RCP [7]). For clarity we refer to such an iBGP-speaking IRSCP as RCP. RCP exchanges routes with PEs using iBGP and never communicates with CEs directly. Suppose that RCP sends an iBGP update to PE *B* containing route *A*. To implement full route control, the update must override any routes that PE *B* receives from other CEs (CE *B*). However, this implies that PE *B* never sends alternative routes (route *B*) to RCP unless route *A* fails (or changes its attributes). RCP is thus deprived from using any but the first route it learns. We conclude that an iBGP-speaking IRSCP restricts the ability to apply full route control.

3.2.1 Distribution

Though logically centralized from a route control application viewpoint, IRSCP is implemented as a distributed system—consisting of multiple IRSCP servers—to address fault-tolerance and scalability requirements. If we designed IRSCP as a single centralized server, failure or partitioning away of that server would leave every

PE in the network steerless and unable to forward traffic correctly, since in BGP a session failure implicitly causes BGP routes announced through the session to be withdrawn. In IRSCP we can tolerate the failure of an IRSCP server by letting routers peer with multiple IRSCP servers. Furthermore, a centralized IRSCP faces a number of (overlapping) scalability challenges. First, a large Tier-1 ISP’s routers collectively maintain many thousands of BGP sessions with routers in neighboring networks, something that no current BGP implementation is able to support by itself. Second, IRSCP makes independent routing decisions for each of the hundreds of PEs within the ISP. Third, IRSCP must store the combined BGP routes received from CEs and originated by the network, and it must process updates to these routes.

As shown in Figure 3, we choose to partition the IRSCP workload by letting each IRSCP server peer with a subset of PEs and CEs, thereby addressing the first two scalability concerns. Our architecture still requires each IRSCP server to store all BGP routes and process the corresponding updates; however, we show in Section 5 that this aspect of scalability does not pose a severe problem.

In performing the per-PE decision process, an IRSCP server needs to take into account the position of that PE in the IGP topology. To maintain consistency of IGP routing state, each IRSCP server runs an IGP viewer and has the same global view of the IGP topology [7]. Due to the partitioning of work in our distributed architecture, each IRSCP server needs to perform shortest-path calculations only for the set of PEs for which it makes BGP routing decisions rather than for the network as a whole. The IRSCP servers further ensure consistency by exchanging *all* BGP updates among each other. Comparing this solution with BGP route reflection [4], the most important difference is that a route reflector selects a *single route* as best route for each destination (using the “normal” BGP decision process) and only makes that route available to other routers and route reflectors. As a result different routers and route reflectors may observe a different set of available routes, which in turn has led to non-deterministic or divergent behavior in BGP, e.g., “MED oscillation” [17, 26].⁴ Basu *et al.* [3] show that exchanging all routes selected by Steps 0-4 of the decision process is sufficient to prevent non-determinism and divergence in iBGP; IRSCP exchanges a superset.

Thus we propose a simple and elegant solution to distribution: a full mesh in which all IRSCP servers distribute all routes. Architecturally, this solution is similar to an iBGP infrastructure in which BGP routers are fully meshed, an architecture that was eventually replaced by route reflectors due to scalability problems. However, there are two differences between the two architectures.

⁴MED oscillation continues to be observed in the Internet [36].

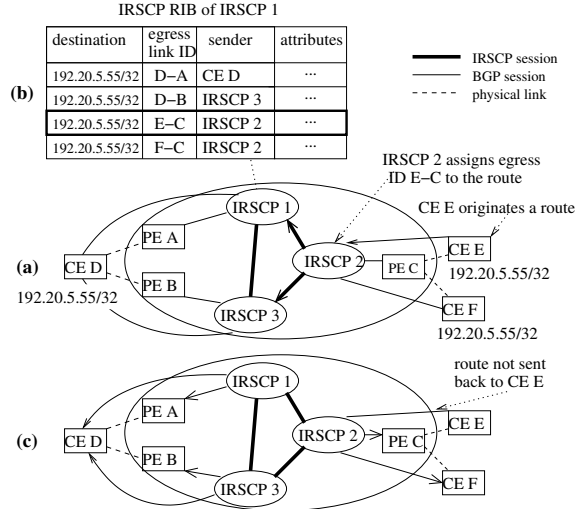


Figure 4: IRSCP route propagation example.

First, the number of IRSCP servers is small compared with the number of routers. Second, being based on commodity technology rather than router technology, we expect IRSCP servers to keep better pace with the technology curve than routers have [20].

3.2.2 IRSCP protocol and RIB

The *IRSCP protocol* (Figure 3(b)) is responsible for ensuring route exchange among IRSCP servers and is implemented as a simple extension to BGP. Each pair of IRSCP servers maintains a TCP-based *IRSCP session* through which they exchange incremental updates in the form of advertisements and withdrawals of routes. At session startup the IRSCP servers exchange advertisements corresponding to all known BGP-learned routes, similar to “normal” BGP sessions. When an IRSCP session goes down, all routes exchanged previously on the session are implicitly withdrawn. When an IRSCP server learns a route from a CE router, it sends the route to all other IRSCP servers through the IRSCP sessions. (For simplicity an IRSCP server does not propagate routes learned from one IRSCP server to another.)

Figure 4 shows an example where the same destination prefix (192.20.5.55/32) is advertised by three different CEs connected to three different PEs. As shown in Figure 4(a), IRSCP 2 learns two routes to destination 192.20.5.55/32 through BGP, and it must send both instances to IRSCP 1 and 3. To distinguish several routes for the same destination, the IRSCP protocol includes in each update an identifier for the egress link to which the route corresponds. The *egress link identifier* (*egress ID* for short) is a (CE,PE) pair of the routers incident on the egress link. For example, from Figure 4 the routes learned through IRSCP 2 have egress IDs *E - C* and *F - C*.

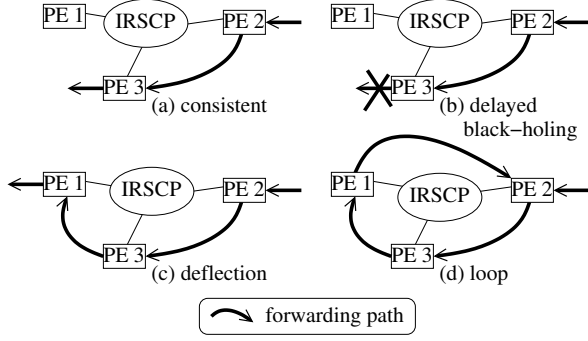


Figure 5: Forwarding anomalies. In all cases the decision process for PE 2 has selected an egress route through PE 3 as the best route for some destination. (a) The decision process for PE 3 has selected a local egress route as best route, and therefore is consistent. (b) The decision process for PE 3 has not selected any route for this destination, thus traffic is black-holed. (c) The decision process for PE 3 has selected PE 1 as best egress route, resulting in a deflection. (d) The forwarding loop is a result of multiple deflections.

When an IRSCP server receives routes from BGP routers and from other IRSCP servers, it stores the routes in a routing information base (RIB), so that the routes are available to the decision process and for further propagation to routers and IRSCP servers. For example, the IRSCP RIB of IRSCP 1 shown in Figure 4(b) contains four entries for prefix 192.20.5.55/32. Each entry has fields for the destination prefix, the egress ID, the neighbor of the IRSCP server from which the route was received, and BGP attributes that belong to the route. In the example, CE D has sent routes to IRSCP 1 and 3, resulting in the first two entries. The last two entries correspond to routes sent to IRSCP 2 by CEs E and F .

Based on the RIB, an IRSCP server executes a decision process and sends a single route per destination (Figure 4(c)) to each attached BGP router. Since the IRSCP server only advertises one route per destination through each BGP session, the egress link ID is not needed (nor recognized by BGP) and so is stripped before sending the route.

3.2.3 Ensuring consistency

The concept of application-provided explicit rankings permits a route control application a great deal of flexibility. However, it also introduces the possibility of IRSCP executing the decision process in an inconsistent manner for different PEs, which can lead to forwarding anomalies. Figure 5 depicts the forwarding anomalies that may result: delayed black-holing, deflection, and forwarding loops. A packet is said to be deflected if a router on its forwarding path chooses to forward to a different egress router than the egress router previously selected by a

router upstream on the forwarding path [18]. In an MPLS network deflections only occur at egress routers. We wish to prevent deflection for two reasons. First, given the existence of shortest-path MPLS tunnels between two PEs, forwarding through an intermediate BGP router is suboptimal (Figure 5(c)). Second, multiple deflections can lead to a forwarding loop (Figure 5(d)). Similarly we wish to avoid delayed black-holing as in Figure 5(b) since it is wasteful to carry traffic through the network only to have it dropped. If the intent is for the traffic to be dropped, it should be dropped on ingress (i.e., at PE 2).

Ultimately, the *correctness* of the rankings is specific to the application. However we consider *consistency* to be a minimum standard of correctness for any route control application and therefore define a set of per-destination constraints on any set of rankings provided by an application. Enforcing these constraints (by an application or by a resolver in a pluggable route control application) ensures that the explicitly ranked decision process is *deflection-free*, i.e., free of deflections and delayed black-holing. More formally:

Definition: *Deflection-free:* For each PE r : if egress route e is selected as best egress route for PE r , then some route f is selected as best egress route for PE $pe(e)$ (the PE incident on e) such that $pe(f) = pe(e)$.

Instantiating PE 2 for r and PE 3 for $pe(e)$ it should be obvious that Deflection-freeness prevents the anomalies shown in Figure 5(b) and (c). In Figure 5-a PE 3 selects *some* route through itself (which is sufficient to prevent an anomaly), although we cannot tell from the figure if PE 3 has chosen e or some other route f for which $pe(f) = pe(e)$.

Claim: The BGP decision process in IRSCP is Deflection-Free.

Proof: Suppose for router r egress route e is selected as best egress route. If $r = pe(e)$ we are done, so assume $r \neq pe(e)$. Since e is in the egress set of r and all routers share the same egress set, e is also in the egress set of $pe(e)$. (Recall the definition of egress set in Table 1.) Note that for every route f : f is an eBGP-learned route from the perspective of $pe(e)$ iff $pe(f) = pe(e)$. Therefore the BGP decision process for $pe(e)$ has at least one eBGP-learned route when it enters Step B-5 and eliminates all non-eBGP-learned routes in Step B-5. Eventually it must select an eBGP-learned route. ■

We define the operator $<_r$ as: $e_1 <_r e_2$ iff in the explicit ranking for router r egress link e_1 is ranked above egress link e_2 . For example, for PE A in Figure 2(a), we have $E - C <_A F - C$ and $F - C <_A G - D$. We noted earlier that Steps R-5 and R-6 of the explicitly ranked decision process effectively extend each explicit ranking to include all egress routes. We say that $e \in <_r$ if e is in the explicit ranking for r . For example $E - C \in <_A$, but if

some new egress D-H appeared then $D - H \notin <_A$. The constraints on explicit ranking are as follows.

Definition: *Ranking-Consistent-1:* The set of egress routes appearing in each router’s explicit ranking is identical.

Definition: *Ranking-Consistent-2:* For each router r and all egress links e_1, e_2 : if $e_1 <_r e_2$ then $e_1 <_{pe(e_1)} e_2$, where $pe(e)$ is the PE incident on e .

The rankings shown in Figure 2(a) clearly satisfy Ranking-Consistent-1: all rankings contain the same egress links. They also satisfy Ranking-Consistent-2. For example, checking the ranking for PE B we see that (1) $G - D <_B E - C$ and $G - D <_D E - C$, (2) $G - D <_B F - C$ and $G - D <_D F - C$, (3) $E - C <_B F - C$ and $E - C <_C F - C$.

Claim: If the explicit rankings given to an explicitly ranked decision process satisfy Ranking-Consistent-1 and Ranking-Consistent-2 then the explicitly ranked decision process is deflection-free.

Proof: Suppose for router r egress route e is selected as best egress route. We show that e is also selected as best egress route for router $pe(e)$. Since e is in the egress set of r and all routers share the same egress set, e is also in the egress set of $pe(e)$. We also know that e is most preferred among the routes in the egress set by Steps R-5 and R-6 of the explicitly ranked decision process for r , and that by Ranking-Consistent-1 the same egress links appear in r and $pe(e)$ ’s explicit rankings. There are two cases: e does or does not appear in the explicit rankings. If e does not appear in the explicit rankings, *none* of the routes in the egress set appear in the explicit rankings (or Step R-5 would have selected a different route for r). Therefore both $pe(e)$ and r identically rank the egress set using Step R-6, and $pe(e)$ selects e . If on the other hand e does appear in the explicit rankings, then r has selected it in Step R-5. Furthermore, with e available we know that $pe(e)$ must select *some* route e_2 that also appears in the explicit rankings (and in the egress set). Suppose $e_2 \neq e$. $e <_r e_2$ or Step R-5 would not have selected e for r . But from Ranking-Consistent-2 it follows that also $e <_{pe(e)} e_2$ and so $pe(e)$ cannot have selected e_2 . ■

Essentially, the ranking abstraction is able to describe a preferred egress link for each PE and per-PE fail-over behavior such that traffic does not get deflected. It is powerful enough to express any consistent assignment of egress routes to routers. However, the constraints do not permit failing over from one arbitrary consistent assignment to another. For example a given set of rankings that ranks egress link e_1 highest for PE A cannot fail over in such a way that egress link e_2 is assigned to PE A , unless e_1 fails.

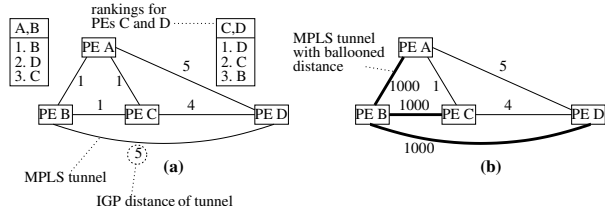


Figure 6: Example of IGP ballooning. All routers are connected by MPLS tunnels whose distance metric is computed by IGP. (a) shows a topology based on which the application has computed a set of rankings. In (b) the distance of various tunnels has increased excessively.

3.2.4 IGP reachability

We assume that the application has taken IGP distances into account when it creates the ranking. Although the IRSCP decision process could conceivably re-rank egress links in response to IGP distances, it generally does not do so for several reasons. First, for applications such as load balancing customer traffic, strict adherence to a shortest-path policy appears to be of secondary importance. Indeed, tracking IGP distance changes can have adverse effects, such as causing large volumes of traffic to shift inadvertently [33]. The explicit ranking provided by an application introduces a degree of stability, effectively “pinning” routes. If it is necessary to respond to IGP changes, we require the application to do so by providing an updated ranking. Teixeira *et al.* [32] suggest that in a large ISP with sufficient path diversity in its IGP topology the latency of MPLS tunnels is not greatly affected by IGP changes. For these cases, route pinning does not sacrifice much performance in terms of latency.

However, we do wish to handle the case in which IGP distances “balloon” excessively, effectively making some egress routes unusable. For example, this can occur when physical connectivity is disrupted and IGP diverts traffic around the disruption. Another example is router maintenance: typically the maintenance procedure involves setting the IGP distance between the router and the rest of the network to a very high value in order to gracefully move the traffic away from the router before it is brought down.

The network shown in Figure 6 has three egress routes for some given destination: through PEs B, C and D . The application has assigned the egress route through PE B to PEs A and B and the egress route through D to PEs C and D . In Figure 6(b) several IGP distances have ballooned, making PE A ’s preferred egress route through B virtually unusable for PE A , although A ’s rankings have not yet been updated.

We define an *Emergency Exit* procedure for cases such as this, which is as follows. If an IRSCP server finds that

the IGP distance from a PE to the PE’s preferred egress route balloons, the IRSCP server ignores the rankings for that PE and destination and reverts to hot-potato routing (i.e., selects the nearest egress router, possibly the PE itself).⁵ In the example, PE *A* overrides its ranking and chooses PE *C*. PE *C*’s most preferred egress route (through *D*) has not ballooned and therefore PE *C* deflects to PE *D*, at which point the traffic egresses.

As this example shows, ignoring the rankings may lead to a deflection. We consider this acceptable, since (a) in a well-engineered network excessive ballooning should be the exception and (b) at most one deflection (and no delayed blackholing) can occur, and therefore no forwarding loop can occur, which we prove below.

Claim: If PE R_1 forwards traffic to PE R_2 , and if R_2 invokes Emergency Exit, then the traffic must egress the network at R_2 (without getting black-holed).

Proof: Since R_1 forwards traffic to R_2 , it must have a route e in its egress set with $pe(e) = R_2$ and which is ranked higher than the black-hole egress ID b (independent of whether R_1 uses the explicitly ranked decision process or Emergency Exit). By the consistency constraints on the rankings and by the fact that all routers share the same egress set, R_2 has e in its egress set and ranks e above b . If R_2 now invokes Emergency Exit, it must select e (or some other local egress route other than b) in Step B-5 and egress the traffic. ■

From this it follows that the explicitly ranked decision process, enhanced with Emergency Exit, does not permit delayed black-holing.

Claim: Enhancing the explicitly ranked decision process with Emergency Exit permits at most one deflection in any forwarding path.

Proof: We consider traffic for some destination d that enters at some ingress PE A and assume to the contrary that two (or more) deflections do occur, i.e., the traffic is forwarded from PE A to some PEs B , C and D , in that order. From the previous claim, we know that B and C do not invoke Emergency Exit, but instead follow the explicitly ranked decision process. Since B and C do not use themselves as the egress, it is clear that they could not have invoked Emergency Exit. Now consider a different (but concurrent) flow of traffic for d , ingressing at B . Since a router’s forwarding behavior does not depend on where the traffic originates, this traffic follows the path $B - C - D$. In other words, this traffic is deflected without passing through a router that invokes Emergency Exit. This implies that the explicitly ranked decision process executed by B and C is not deflection-free, a contradiction. ■

⁵However, if a black-hole egress ID is present in the ranking, then IRSCP still excludes egress routes that are ranked below the black-hole egress route before executing the hot-potato steps.

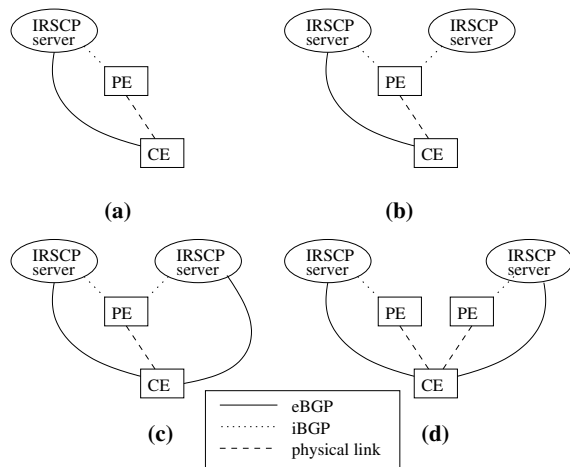


Figure 7: Replicating customer connectivity. (a) No replication. (b) Replication of PE-IRSCP connectivity. (c) Replication of the CE-IRSCP eBGP session. (d) Replication of all components that make up customer connectivity.

3.2.5 Fault tolerance

We now discuss how IRSCP handles a number of common failure scenarios including the loss of customer connectivity to IRSCP, failure of individual IRSCP sessions and IGP failures, and route-control application communication failures.

First we examine **failure in customer connectivity to IRSCP**, by which we mean the ability for a CE to announce or learn reachability of a route to or from IRSCP. As is apparent from Figure 7-a, CE connectivity can be disrupted by failure of an IRSCP server, a PE, an eBGP session, an iBGP session or a physical link. (We consider failure of the CE the customer’s responsibility.) Robustness to failure of any of these components can be improved by having the customer connect to several PEs and several IRSCP servers, as shown in Figure 7-d. In addition, individual components can be made more robust as shown in Figure 7-b and c.

Since IRSCP servers do not re-advertise updates learned from other IRSCP servers, each IRSCP server can only learn a particular route from one IRSCP server (i.e., the IRSCP server that learned the route via an eBGP session with a router). A failure of an IRSCP session can therefore cause IRSCP servers to learn a different set of routes, potentially leading to inconsistency. Our aim is to prevent inconsistency from leading to a situation in which (a) IRSCP servers send updates to a PE that are inconsistent with updates sent to other PEs, (b) several IRSCP servers send inconsistent updates to the same PE. Among many potential causes underlying an IRSCP session failure we focus on misconfiguration of IRSCP peering and network partitioning.

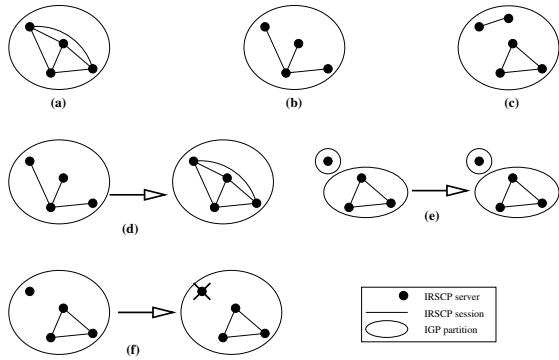


Figure 8: Healing the IRSCP graph within a network partition. (a) Complete IRSCP graph. (b) Incomplete IRSCP graph. (c) Partitioned IRSCP graph. (d) Adding missing IRSCP peerings to an incomplete IRSCP graph. (e) A network partitioning. (f) Partitioned IRSCP graph in which all but one partition consists of a single IRSCP server.

We first discuss the case of a **network (IGP) partitioning**. By definition, a PE cannot have connectivity to IRSCP servers in different network partitions. Furthermore, there is no reachability between PEs in different network partitions and therefore no inconsistent routing between such PEs. It follows that it is sufficient to ensure consistency among the IRSCP servers within each network partition separately.

Next we examine inconsistency within a network partition. We define an *IRSCP graph* consisting of IRSCP servers (vertices) and IRSCP sessions (edges) within a single network partition. In the absence of failure of IRSCP sessions within the network partition the IRSCP graph is complete and consists of a single connected component (Figure 8-a). **Failures of IRSCP sessions** within the IGP partition can cause the IRSCP graph to become incomplete (Figure 8-b) or even partitioned (Figure 8-c). We define procedures that “align” each IRSCP graph partition with the corresponding network partition and “heal” incomplete IRSCP graphs.

To handle failures within one partition in the IRSCP graph, we have IRSCP servers signal to each other (through the IRSCP sessions) which IRSCP servers are in their partition. If an IRSCP server I notices that it does not have an IRSCP peering with some IRSCP server J in its IRSCP partition, it establishes the peering with J (Figure 8-d). For failures that partition the IRSCP graph, we consider two sub cases. The case shown in Figure 8-c in which multiple IRSCP graph partitions contain more than one IRSCP server we assume to be highly unlikely, since it implies that all IRSCP servers in every such partition are misconfigured, and so we do not handle this case. If on the other hand an IRSCP server finds itself *alone* in an IRSCP graph partition (i.e., it cannot estab-

lish any IRSCP sessions), it proceeds by checking its IGP viewer to see if any of the IRSCP servers it is trying to contact are present in its IGP partition. If none of them are, the IRSCP graph partition is in fact a network partition (Figure 8-e), and the IRSCP server continues to function correctly (see above). If the IRSCP server does find one of the other IRSCP servers in its network partition (Figure 8-f) it is highly likely that there is a problem with the IRSCP server itself. Rather than introducing potential inconsistencies, the IRSCP server halts. Here we rely on replication of IRSCP servers (discussed earlier) and let another IRSCP server take over.

Finally we look at failures related to the **the route control application**, specifically (a) omissions of PEs from the rankings generated by the application, and (b) failure of communication between the route control application and one or more IRSCP servers. When a PE is added to the network, the application must become aware of it so that the application can generate a corresponding per-PE ranking. However, relying on configuration is error-prone and may lead to persistent omissions of PEs from the rankings, potentially leading to inconsistency. Our solution is to rely on the Emergency Exit procedure (Section 3.2.4): an IRSCP server that needs to execute the decision process for a PE for which it has not received ranking reverts to the BGP decision process. Similar to the Emergency Exit procedure this may lead to at most one deflection, and cannot lead to a forwarding loop.

When communication between the route control application and one or more IRSCP servers fails, some IRSCP servers may be excluded from ranking updates while others are not. A special case of this failure mode is that the route control application as a whole fails or is separated from IRSCP. With the rankings inconsistent the explicitly ranked decision process may no longer be Deflection-Free. To handle these failures, we proceed as follows. First, with each newly generated set of rankings (rankings concerning all PEs and all destinations), the application uploads the new ranking set to each IRSCP server, together with a version number that identifies the ranking set. (Of course, to reduce communication cost the application can transmit successive ranking sets using incremental updates.) Above, we described that IRSCP servers inform each other as to what IRSCP servers are present in their IRSCP graph partition. In this communication each IRSCP server ID is accompanied by the version number of the ranking set used by that IRSCP server. When an IRSCP server notices that it is no longer receiving rankings from the application it compares its version number with the version numbers of other IRSCP servers in the IRSCP graph partition. If it finds that some other IRSCP server has a more recent ranking set, it requests that ranking set from the other IRSCP server. As a result eventually all IRSCP servers in an IRSCP graph partition

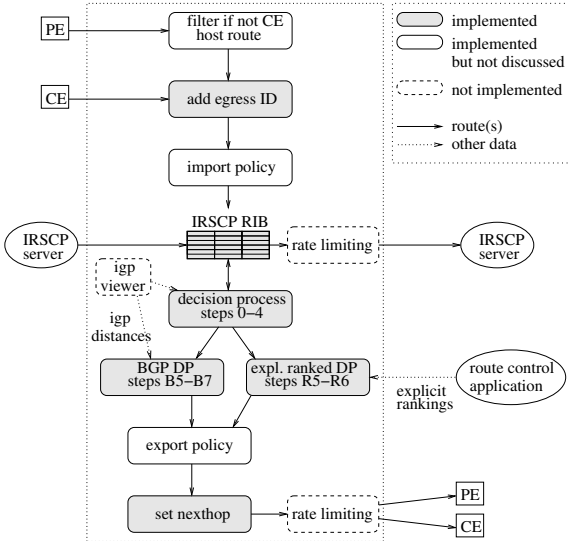


Figure 9: Control flow through an IRSCP server.

observe the same ranking set. Since as discussed each IRSCP graph partition is aligned with a network partition, it follows that eventually all IRSCP servers in the same network partition have the same ranking set.

4 Implementation

A significant part of the IRSCP server functionality is identical to that of a BGP router. We therefore based our prototype implementation on the version 3.9 code base of `openbgpd`.

Figure 9 summarizes the control flow among the various components that our implementation uses to receive, store, process and send routes. The main high-level addition with respect to our earlier discussion is that routes exchanged with BGP routers (PEs and CEs) must undergo processing after being received and before being sent. With the aid of Figure 9, we now discuss the implementation of a number of specific IRSCP functions in detail. Note that while we have implemented an IGP viewer for a previous centralized control platform [7], we have not yet ported it to our IRSCP prototype. In addition, BGP defines a rate limiting mechanism (“MinRouteAdvertisementIntervalTimer” [29]) that we have yet to implement for BGP sessions and adapt for IRSCP sessions.

4.1 Explicit application ranking

As shown in Figure 9, route control applications provide rankings for the IRSCP’s explicitly ranked decision process. In our prototype they do so through the IRSCP configuration file, which contains a number of `rank` statements, one for each set of prefixes that are ranked identi-

cally. The following is an example of a `rank` statement for two prefixes.

```
rank {
  prefix { 3.0.0.0/8, 4.0.0.0/8 }
  pe { 1.2.3.4, 5.6.7.8 }
  egresses { 11.12.13.14:15.16.17.18,
             19.20.21.22:23.24.25.26 }
  pe { 101.102.103.104 }
  egresses { 19.20.21.22:23.24.25.26,blackhole }
}
```

The `pe` statements of a `rank` statement must contain every PE attached to the IRSCP server. Recall that in IRSCP each route carries an egress ID, specifying what egress link traffic for the destination of the route is to use when it exits the network (assuming the route is selected as best route). The `egresses` statement is the ranking of egress IDs to be used for the PEs in the preceding `pe` statement. Each egress ID consists of the IP addresses of the CE and PE incident on the egress link. In addition, the `blackhole` egress may be specified. Application rankings are stored in a per-PE Red-Black Tree of destination prefixes, where each prefix points to the ranked list of egress IDs for that PE and prefix. When a new ranking set arrives, the IRSCP server updates its ranking tree and reruns the decision process for affected destinations.

4.2 Decision process and IRSCP RIB

The data structures used to implement the RIB (*IRSCP RIB* in Figure 9) are adapted from `openbgpd`’s implementation of the BGP RIB. `openbgpd` provides various indexes into the BGP RIB, the most important of which is a Red-Black Tree of destination prefixes, where each entry points to a list of routes (one route for each neighbor). To simplify the implementation we maintain this structure, despite the fact that the number of routes per prefix in IRSCP increases to one route per egress link. Our performance evaluation in Section 5 shows that the resulting increase in search time for a route in the RIB is not a great concern.

`openbgpd` maintains the per-destination list of routes in order of preference, based on pairwise comparison of the BGP attributes of routes according to the BGP decision process steps shown in Table 1. Thus each route update is linear in the number of routes for the destination, and the decision process itself amounts to no more than checking whether the egress router for the route at the head of the list is reachable (Step 0).

However, there are two problems with this algorithm for IRSCP. First, pairwise comparison of the MED value (Step 4) produces results that may be incorrect and, furthermore, dependent on the order in which the routes were inserted into the RIB, which in turn leads to po-

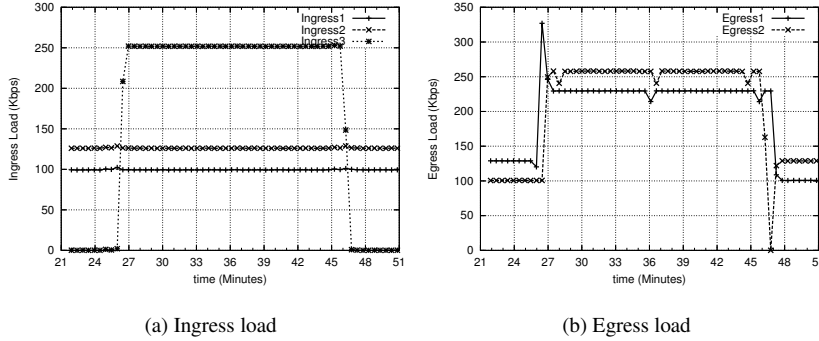


Figure 10: Functional evaluation.

tential inconsistency among the RIBs in different IRSCP servers. The underlying problem is that MED, being comparable only between routes from the same neighboring network, does not follow the “rule of independent ranking” [17].

The second problem is that IRSCP defines a per-PE decision process, resulting in a different order of routes for different PEs. However, we note that Steps 0–4 of the (BGP or explicitly ranked) decision process (which compute the egress set) are independent of the PE, hence we compute these steps once for all PEs (*decision process steps 0–4* in Figure 9), storing the result in the RIB. Our implementation orders routes in the RIB using pairwise comparison based on Steps 0–3. Next, of the routes ranked highest so far, it examines each set of routes received from the same neighboring network and sets a flag on those that have highest MED value in such a set, thereby computing the egress set (Step 4). The algorithm used for Step 4 is similar to that used by the Quagga open source BGP stack (www.quagga.net) and runs in $O(n^2)$ time for n routes available for a prefix.

Next, the PE-specific steps of the decision process are executed using a pairwise comparison of routes in the egress set. In the case of the BGP decision process (*BGP DP steps B5–B7* in Figure 9), we break the tie in Step B-7 based on the egress ID of the route rather than the router ID (of the BGP router or IRSCP server that sent the route), since a neighboring IRSCP server may have sent multiple routes for the destination. In the case of the explicitly ranked decision process (*expl. ranked DP steps R5–R6* in Figure 9), the IRSCP server adds a black-hole route (with lowest possible egress ID) to the egress set and then retrieves the list of ranked egress IDs for the PE and destination (Section 4.1). When comparing the egress IDs of a pair of routes in the egress set, our implementation simply walks down the list of egress IDs until it finds the higher ranked of the two egress IDs. This runs in time linear in the number of routes times the size of the ranking for a prefix and PE, or, if all n prefix’s routes appear in its ranking, in $O(n^2)$ time.

Following (re-)execution of the decision process for a given PE, the IRSCP server distributes its decision to the

PE and to all CEs attached to the PE. Note that the IRSCP server does not need to run a separate decision process for a CE: as in BGP, the route sent to the CE is the same as is selected for the associated PE.⁶ To prevent unnecessary updates from being sent to neighbors, BGP implementations typically remember the last route selected for each destination as the *active route*. Our IRSCP server implementation uses the same technique; however, instead of storing a single active route, it stores an active route for each attached PE.

To evaluate scaling of the decision process in terms of the number of routing policies, we should consider not only the time needed to evaluate the ranking of a prefix (linear time, see above), but also the time needed to look up the ranking. Retrieving a ranking runs in time logarithmic in the number of ranked prefixes and (in our implementation) linear in the number of PEs per IRSCP server. This is similar to the time needed by a BGP implementation to look up a prefix in the routing table and retrieving peer state before sending updates to a peer.

4.3 Egress management

Based on the egress ID specified in each IRSCP route, IRSCP has to ensure that (a) BGP routers are “instructed” to forward the traffic towards and through the indicated egress link, and (b) the route is only used if the egress link is available. To implement (a) we use the *next-hop* BGP attribute of a route. This attribute tells a router to which *next-hop router* it must forward traffic when it uses the route. The IRSCP server sets the *next-hop* attribute (*set next-hop* in Figure 9) when it sends a route to a PE or CE in such a way that traffic passes through the egress link and uses the egress ID to determine what that *next-hop* should be. For example in Figure 2(a) and (b), IRSCP determines from egress ID $E - C$ that PE A must use PE C as *next-hop*, and PE C must use CE E as *next-hop*. In addition, a CE attached to PE A (not shown) is sent PE A as *next-hop*. The latter is not determined based on egress ID. Rather, as we discuss next, an

⁶Ignoring the fact that eBGP export policies may still be applied.

IRSCP server associates each CE with a PE, and it is this PE that IRSCP sets as nexthop when sending to the CE.

When an IRSCP server is configured to form an eBGP session with a CE, part of the configuration identifies a PE with which to associate the CE: the PE to which the CE is physically attached through an egress link (e.g., see Figure 3(b)). Apart from setting the nexthop router for routes sent to the CE (as discussed above), the IRSCP server uses the identity of the PE and CE to set the egress ID (*add egress ID* in Figure 9) on routes received from the CE.

To implement (b) above, the IRSCP server monitors the validity of the egress link and thereby of any routes received from the CE. Normally when two BGP routers exchange routes through a BGP session, the validity of the routes are preconditioned on the liveness of the session: if the session goes down, it is assumed that the physical links that carry the session’s packets are unusable and any routes exchanged must be withdrawn.

However, in the case of a multihomed customer it is possible that the eBGP session’s packets are routed through any of the customer’s egress links.⁷ Therefore an IRSCP server explicitly makes an eBGP session dependent on the correct egress link by checking whether an *egress status route* was received from the PE. The egress status route is a BGP route for the address of the CE, configured on the PE, and made dependent on the presence of the egress link. The IRSCP does not allow the eBGP session to be formed unless the egress status route for that session is present.⁸ Note that in the IRSCP architecture presented in Section 3, an IRSCP server normally does not receive routes from PEs. The egress status route is an exception, and our IRSCP implementation guards against configuration errors by ensuring that only host routes are received from PEs (*verify egress route* in Figure 9).

4.4 Import and export policy

IRSCP provides a convenient interface for controlling the flow of traffic through an ISP’s network. However it must also accommodate today’s routing policies, e.g., controlled distribution of routes to customer vs. peering networks, manipulating BGP attributes such as AS paths and MED values, and filtering undesired routes from neighboring networks. Although ideally IRSCP should enable a centralized configuration of routing policy through an application, we leave this aspect of route control as future

⁷In conventional BGP, it is considered good practice to only configure eBGP sessions between routers that share a common link, namely the link used by traffic for destinations exchanged on the session, and to restrict the session’s packets to use that same link.

⁸Egress status routes also allow the large number CE IP addresses to be kept outside of the ISP’s IGP. Some ISPs prefer to minimize the number of IP addresses carried by their IGP.

work. In the meantime, it is important for IRSCP to support today’s style of routing policy configuration without introducing inconsistency into the decision making.

To ensure consistency we allow policy to be applied only after receiving a route from a CE (*import policy*) or before sending a route to a CE (*export policy*). Applying policy at these points can never hurt consistency, since the effects are identical to the CE applying a local export or import policy, respectively. We believe that the combination of IRSCP ranking and CE-import and CE-export policy enables virtually all policy in use today. For example, all import and export policies discussed by Caesar et. al. [8] are configured for routes entering or leaving the network. One class of policies that we do not allow are those that have a per-PE (as opposed to network-wide) effect. Examples of mechanisms that implement such policies are Cisco’s weight and cost community [11, 30] attributes. A similar effect can be achieved in IRSCP using an explicit ranking.

5 Evaluation

In this section we first present a functional evaluation, demonstrating that an example load-balancing application is able to effect fine-grained route control using the ranking abstraction and our prototype IRSCP implementation. We then evaluate the scalability and performance of our prototype in processing BGP updates.

5.1 Functional evaluation

We verified the functionality of IRSCP with a testbed consisting of an IRSCP server, a load-balancing route control application, three ingress PEs, (Ingress1, Ingress2, and Ingress3), two egress PEs, (Egress1 and Egress2), a core router, and a number of hosts arranged into a “source” network and a “destination” network. The core router uses point-to-point links to connect to the PEs and we use GRE tunnels as the tunneling technology between the PEs. We assume a simple scenario in which the egress PEs attach to a customer who prefers its incoming traffic to be balanced on the egress links.

The load-balancing application performs SNMP GET queries on each PE to collect offered and egress loads. Based on the offered load, the application computes a ranking set that balances the load on the egress links. If the ranking set changes, the application generates an updated configuration and issues a configuration reload command. The hosts in the source network send constant rate UDP packets to the hosts in the destination network. Figure 10(a) shows the offered load at the ingress PEs as measured by the application server.

The load at the egress PEs is shown in Figure 10(b).

Before $t = 26$, the load at these two egresses is “balanced” as a result of the initial ranking set at the IRSCP (i.e., Ingress1 prefers Egress2, and Ingress2 prefers Egress1). At $t = 26$, the offered load at Ingress3 increases, and the application takes up to 30 seconds (the SNMP polling interval) to detect this change. It then realizes that the best way to balance the load is to send the load from Egress1 and Egress2 to one egress point, and the load from Egress3 to another. Therefore, it generates a new ranking set and sends it to the IRSCP server, which updates the ingress PEs. As a result, the load from Ingress3 goes to Egress2, and the load from Ingress1 and Ingress2 goes to Egress1. Similarly, at $t = 47$ the application generates a new ranking set and shifts Ingress1 and Ingress2’s loads to different egresses.

5.2 Performance measurement testbed

Rather than emulating an entire ISP infrastructure, we run our tests on a single IRSCP server, loading it as though it were part of a hypothetical Tier-1 ISP network. Note that there is limited value in emulating an IRSCP-based infrastructure for the purpose of performance evaluation. Due to the fact that IRSCP servers exchange all routes, routing in an IRSCP infrastructure does not oscillate like it does in a BGP-based infrastructure [3]. (Whereas BGP routers and route reflectors exchange only their best routes, IRSCP servers exchange all eBGP-learned updates with each other, and so an IRSCP server does not change its route selections for a destination once it has received all such routes.) Therefore convergence time of an update is governed solely by communication latency and the processing latency of an update in an IRSCP server or a BGP router.

Figure 12(a) shows the Tier-1 ISP network modeled. Each of the n_{pop} POPs (Point-of-Presence, e.g., a city) contains one IRSCP server and all IRSCP servers are fully meshed. Assuming IRSCP as a whole processes a combined BGP update⁹ rate $rate_{all}$ from all CEs, then on average an IRSCP server sends roughly $rate_{all}/n_{pop}$ to each IRSCP server, namely the share of updates that it receives from CEs in its own POP. In other words, each IRSCP server sends and receives at a rate of about $\frac{n_{pop}-1}{n_{pop}} \cdot rate_{all} \approx rate_{all}$ on its combined IRSCP sessions (Figure 12(b)). The update rate that an IRSCP server sends to a PE or CE is governed by the output of best route selection based on the (BGP or explicitly ranked) decision process. We make the simplifying assumption that this rate $rate_{best}$ is the same for all PEs and CEs. From measurements taken in a Tier-1 ISP network we derive highly conservative ball park estimates for $rate_{all}$ and $rate_{best}$ (see next section).

⁹By “update” we mean a route update, rather than an update message, which may contain several route updates.

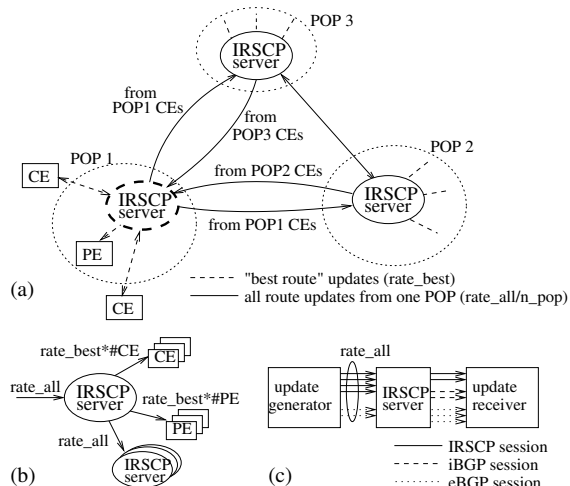


Figure 12: (a) Communication peers of IRSCP server under test. (b) Update rates into and out of the IRSCP server. (c) Experimental setup.

Figure 12(c) shows our experimental setup, consisting of an update generator, the IRSCP server under test and an update receiver. The IRSCP server and update receiver are 3.6-GHz Xeon platforms configured with 4 GB of memory and a 1-Gb Intel PRO/1000MT Ethernet card. The update generator contains a 993-MHz Intel Pentium III processor, 2 GB of memory, and a 100-Mb 3Com Ethernet card. All three machines run OpenBSD 3.8.¹⁰ The three hosts are connected through a Gb switched VLAN, and we monitor their communication by running `tcpdump` on a span port on a separate host (not shown). The update generator and receiver are based on an IRSCP server implementation, with the update generator’s implementation modified to send updates at a desired rate. We configure our setup to only permit updates to be sent from the generator to the IRSCP server and from there to the receiver.

After being loaded with a routing table of 234,000 prefixes, the update generator randomly picks prefixes from the routing table to produce the updates. To ensure that the IRSCP server’s RIB contains a sufficient number of routes per prefix to choose from, the update generator and the IRSCP server maintain 26 sessions. Each time the update generator sends an update message it picks a different session through which to send the message (using round-robin). The message modifies a BGP attribute (the “aggregator” attribute) of the prefixes contained in the message.

The IRSCP server and the update receiver maintain 40 IRSCP sessions, 15 iBGP sessions, and 240 eBGP sessions, reflecting an ISP network consisting of 40 POPs, each of which contains 15 PEs and has sessions with 240

¹⁰The machines are actually dual-processor, but OpenBSD uses only one of the processors.

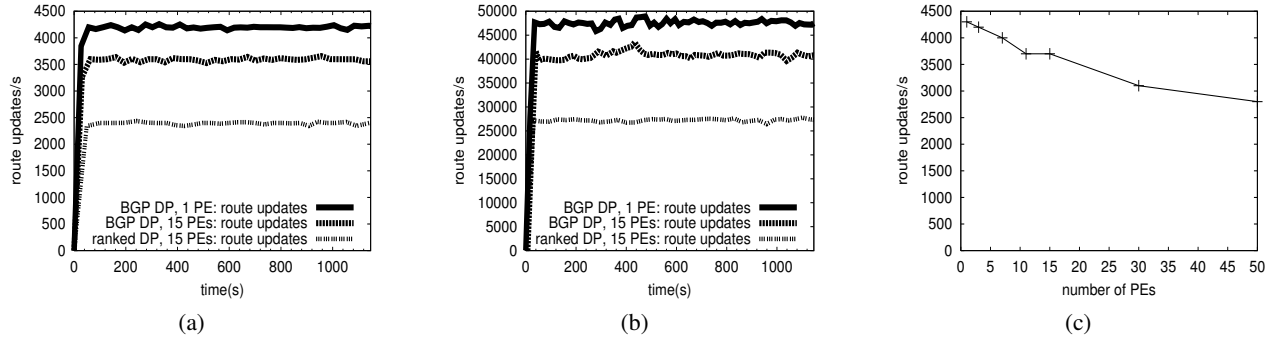


Figure 11: IRSCP server throughput. (a) Sustained input rate: from update generator to IRSCP server. (b) Sustained output rate: from IRSCP server to update receiver. (c) Input rate for varying number of PEs sustained for 40 seconds.

CEs. (Alternatively, this scenario reflects a network consisting of 20 POPs of twice the number of routers, but where each POP contains two IRSCP servers that assume responsibility for half the POP.) We need to ensure that each BGP and IRSCP session with the update receiver sends at a rate of $rate_{best}$ and $rate_{all}/n_{pop}$, resp. Each per-PE decision process in the IRSCP server prefers one of the 26 routes available. Only when the attribute of the best route is changed does the IRSCP server send an update on the iBGP sessions for that PE (and on the eBGP sessions for the associated CEs). By virtue of route selection in the IRSCP server each output BGP session receives a $1/26$ fraction of $rate_{all}$, which turns out to correspond to $rate_{best}$ (see below). We configure one of the 26 input update-generator sessions as an eBGP session and the remainder as IRSCP sessions, ensuring a rate of $rate_{all}/26 > rate_{all}/n_{pop}$ on each output IRSCP session. Finally we instrument the IRSCP server and update generator to send at most three route updates per update message, reflecting the average number of updates per message observed at a route collector in the Tier-1 ISP’s network.

We disabled updates to the kernel forwarding table as well as `openbgpd`’s support for kernel packet filtering, neither of which are required by IRSCP. In addition we prevented `openbgpd` from disabling Nagle’s algorithm (`TCP_NODELAY`) on the peering sessions to achieve better throughput. Before starting each experiment we allowed the routing table to be propagated to the IRSCP server and the update receiver, through each of the sessions.

5.3 Workload used for evaluation

In order to place a reasonable workload on the IRSCP server under test we establish some ball-park upper bounds on the workload of an IRSCP server in deployment, using data collected from a large Tier-1 ISP network. Specifically we estimate upper bounds for the following two items: (a) number of prefixes and routes

carried by an IRSCP server and (b) update rate sent and received by an IRSCP server to and from routers and other IRSCP servers. For (a) we need to count the number of routes and prefixes that are announced by neighboring (customer and peering provider) networks and customer routes originated by the ISP on behalf of customers. We count the number of routes contributed by peering provider networks using BGP routing table dumps taken from the PE routers that maintain BGP sessions with these peering providers. For routes contributed by (or originated on behalf of) customers (i.e. the *customer routes*) we do not have access to corresponding BGP table dumps. However we do have access to *forwarding* table dumps of the PE routers connected to customers, which give us for each customer the set of destinations forwarded to the customer, i.e., the customer’s prefixes. In addition we have a per-customer list of interfaces through which the customer connects to the ISP. We make the conservative assumption that a customer announces each of its prefixes through each of its interfaces, producing an upper bound on the number of customer routes carried by the ISP. Taking the peering provider and customer routes together we arrive at the upper bound estimate of 5,511,277 routes for 214,417 prefixes or 25.7 routes per prefix on average for the month of June 2006.

For (b) ideally we would count the BGP updates received on eBGP sessions by each PE in the Tier-1 ISP network. Unfortunately we do not have access to this information, and therefore we make an estimate based on updates received by a route collector in the Tier-1’s network. First we note that from the perspective of a router (or a route reflector) a destination has at most one best (most preferred) route at all times, though the identity of the best route may change over time. Such a best route corresponds to some external (CE-facing) network interface in the ISP that we call the “best interface” for the destination (and again, whose identity changes over time). Since a router announces its best routes to the ISP’s route collector, when we can examine the update

rate received by the collector from the router for a given destination, we see the update rate corresponding to the best interface for the destination (for that router). Now making the simplifying assumption that each of the destination’s external interfaces produces the same update rate, we have obtained an estimate of the update rate for any single interface for that destination based on that router. However, making this assumption causes some variance in the estimator across different routers, since different routers prefer different interfaces for a destination, and in reality these different interfaces do produce a different update rate. Therefore, we take the average of the estimates across all routers and route reflectors that feed the ISP’s route collector, thereby reducing the variance of the estimator for per-destination, per-interface update rate. (In addition we note that we have ignored the effect of network-internal events such as IGP changes and route collector session resets. However these only make our estimator more conservative.) Now we use our overestimate on the number of routes per destination obtained under (a) previously and multiply that with our per-interface update rate to obtain the estimated upper bound on the total per-destination update rate observed by a deployed IRSCP server. Adding these estimates across all destinations gives the estimated upper bound on update rate to be processed by an IRSCP server. The route collector bins the updates received from its feeds by 15-minute intervals. We therefore base our estimates on the average update rates per time-bin. On the busiest day of the past year in May 2006 (busiest in terms of total updates received by the collector on a day), our estimator gives a maximum total update rate $rate_{all}$ of 4856 updates per second, and a 95th percentile and median value of 627 and 55, respectively.

We can also derive corresponding estimates for update rates sent by an IRSCP server to other IRSCP servers and to PEs and CEs. Given n_i deployed IRSCP servers each connected to a disjoint set of PEs and CEs (e.g., in a deployment with one IRSCP server per POP n_i is just the number of POPs), on average each IRSCP server sends at a rate $rate_{all}/n_i$ to every other IRSCP server. Finally, an IRSCP server sends to each connected PE or CE the best route for each prefix. We estimate this rate $rate_{best}$ as $rate_{all}/avg.\#routes - per - prefix$ giving a maximum rate of 189, and 95th percentile and median values of 24 and 2.1 (on the busiest day).

5.4 Throughput

We determine the maximum value of input rate ($rate_{all}$) that the IRSCP server can sustain for an extended period of time, as follows. To discover the maximum input rate, we gradually increase the input rate and compare the observed output rate with an expected output rate of

$rate_{all}/26 \cdot (n_{irscp} + n_{ibgp} + n_{ebgp})$. When the input rate is below its maximum, the output rate corresponds to the expected output rate. Once the input rate exceeds its maximum, the output rate steadily declines with increasing input rate.

Using this procedure we compare the performance of the standard BGP decision process with the explicitly ranked decision process, and evaluate the impact of the per-PE decision process. For the standard BGP decision process we find a maximum $rate_{all} \approx 3600$ updates/s, corresponding to an expected output rate of 40,846 updates/s. Figures 11(a) and (b) (*BGP DP, 15 PEs*) show the observed input and output rates during a run lasting twenty minutes, averaged over 30-second intervals. While the output rate is not as stable as the input rate, on average it corresponds to the expected output rate, and the figure shows that it is sustainable. Next, we load explicit rankings for 60,000 prefixes, each listing 26 egress IDs in some arbitrary order. In this case we find a maximum $rate_{all} \approx 2400$ updates/s, corresponding to an expected output rate of 27,231 updates/s. As expected, the explicitly ranked decision process is slower than the BGP decision process, since it runs in time quadratic rather than linear in the number of routes per prefix. Again, Figure 11 (*ranked DP, 15 PEs*) shows that the IRSCP server can sustain this workload.

Finally, to evaluate the impact of the per-PE decision process (vs. a single, router-wide decision process), we run an experiment again based on the BGP decision process, but in which all but one of the iBGP sessions between the IRSCP server and the update receiver are replaced by an eBGP session. In this case we find that the IRSCP server sustains a maximum $rate_{all} \approx 4200$ updates/s and produces the expected output rate of 47,654 updates/s (*BGP DP, 1 PE* in Figure 11). Figure 11(c) plots $rate_{all}$ for a varying number of PEs (and using the BGP decision process), but evaluated by sustaining the rate for only 40 seconds, rather than 20 minutes.

While the sustained throughputs are less than our maximum measurement-based estimate for $rate_{all}$ of 4900 updates/s, the IRSCP server easily manages to keep up with our 95th percentile estimate of 600 updates/s in all experiments, even when increasing the number of PEs to 50. Hence, we expect that an improved implementation of flow control in the IRSCP server should sufficiently slow down senders during times that the offered load exceeds the maximum sustainable throughput [21].

5.5 Memory consumption

We evaluate memory usage of an IRSCP server as a function of the number of routes it stores. We use a subset of the setup in Figure 12(c): the IRSCP server under test and the update generator. Also in this experiment we add

BGP attributes from a routing table in the ISP network from October 2006. We vary the number of sessions between the update generator and the IRSCP server from 0 to 20 and measure the memory usage of the IRSCP server’s RIB. We find a linear increase from 18.2 MB (0 sessions) to 226 MB.

Next, we examine memory usage of the explicit rankings. We configure an IRSCP server with rankings for 15 PEs and vary the number of ranked prefixes from 0 to 130,000 (but without loading their routes). Each ranking for a prefix and PE consists of 26 egress IDs. We measure the process size of `openbgpd`’s route decision engine, which stores the rankings. Again we find a linear increase from 840 KB (0 prefixes) to 994 MB (130,000 prefixes). Our current implementation is not optimized for space allocation, and, as a result cannot store rankings of this size for more than 130,000 prefixes. After we load 26 copies of the routing table, our prototype supports rankings for up to 75,000 prefixes on our test machines. However, we expect 26 egress IDs per prefix to be vastly more than needed in practice; five egress IDs per prefix can be stored for all 234,000 prefixes with full routing tables loaded.

6 Related work

IRSCP extends our previous work on route control architectures [7, 13, 34] in three important ways. First, we introduce a modified BGP decision process which we expose to route control applications. Second, IRSCP has *full visibility* of all routes available to the network through its eBGP interaction with neighboring networks. This is in contrast to a phase-one, iBGP-only RCP [7], where routers in the IRSCP-enabled network only pass *selected* routes to IRSCP. Having full route visibility prevents route oscillations within the network [3, 17, 26] and simplifies route control applications. Third, IRSCP distributes the route selection functionality, whereas the simple server replication presented in [7] required each replica to scale to accommodate the entire network. Earlier work on route servers [19] proposed changes to the way routes were distributed between routers, but specifically did not envision any route selection to be performed in these servers. Later, eBGP-speaking route servers [15] similarly addressed the full-meshed connectivity problem between eBGP speakers (typically in Internet exchange points). Bonaventure *et al.* [5] propose sophisticated route reflectors but limit their changes to the iBGP infrastructure. More recent work presented to the IETF proposes to extend the BGP protocol to allow multiple paths to the same destination to be exchanged between two BGP speakers [35]. This is similar to the BGP extensions we implemented to facilitate communication be-

tween the distributed IRSCP servers in our implementation. The 4D project proposes a refactoring of the network architecture, creating a logically centralized control plane separate from forwarding elements [16, 37].

The IETF ForCES working group examines a separation of control plane and forwarding plane functions in IP network elements [23]. This work has a much narrower focus on defining the interface between control and forwarding elements, without considering interaction between different control plane elements. Once ForCES-enabled routers become available, IRSCP’s iBGP communication with local routers might conceivably be replaced by a ForCES protocol.

An earlier IETF proposal [11, 30] provides limited route control by defining a new BGP attribute subtype, the cost community, which can be assigned to routes and used to break ties at a certain “point of insertion” in the BGP decision process. This proposal does not indicate under what conditions the cost community would be safe to use; by contrast, we show how our rankings should be constrained to ensure consistency.

The load-balancing application is a specific instance of traffic engineering, a well-studied problem. For example, the inadequacies of hot-potato routing are also addressed in [32]. The authors propose a TIE ranking metric, which allows operators to trade off reacting to network changes (like hot-potato routing does) versus a more static ranking, which might be designed to favor specific applications or services. Bressoud *et al.* [6] consider the optimal assignment of routes to routers to satisfy both traffic engineering and capacity constraints. Neither of these works fully deal with realization issues, and one can think of these solutions as potential route-control applications for our platform.

7 Conclusion

The ultimate success of a logically centralized control plane architecture will depend not only on its ability to enable new functionality, but also on its ability to provide a scalable and robust routing function to large and growing provider networks. We address each of these points by presenting a distributed realization of the Intelligent Route Service Control Point (IRSCP) that partitions work between instances and allows redundancy requirements to drive the extent to which the system is replicated. We also move beyond BGP capabilities by allowing route control applications to directly influence the route selection process by providing a ranking of egress links on a per-destination and per-router basis. We demonstrate the utility of this change with a simple load-balancing application. As future work, we plan to investigate to what extent an IRSCP that has complete control

and visibility allows a simplification of the expression and management of conventional routing policies.

Acknowledgments. We would like to thank Yuvraj Agarwal, Michelle Panik, Barath Raghavan, Rangarajan Vasudevan, Michael Vrable and the anonymous reviewers for their helpful comments on previous versions of the paper. This work was supported in part by the National Science Foundation through grant CNS-0347949.

References

- [1] Aditya Akella, Jeffrey Pang, Anees Shaikh, Bruce Maggs, and Srinivasan Seshan. A Comparison of Overlay Routing and Multihoming Route Control. In *ACM SIGCOMM*, September 2004.
- [2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *ACM SOSP*, pages 131–145, October 2001.
- [3] Anindya Basu, Chih-Hao Luke Ong, April Rasala, F. Bruce Shepherd, and Gordon Wilfong. Route Oscillations in I-BGP with Route Reflection. In *ACM SIGCOMM*, pages 235–247, 2002.
- [4] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. RFC 2796, April 2000.
- [5] O. Bonaventure, S. Uhlig, and B. Quoitin. The case for more versatile BGP Route Reflectors. Internet draft: draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.
- [6] T.C. Bressoud, R. Rastogi, and M.A. Smith. Optimal Configuration for BGP Route Selection. In *IEEE INFOCOM*, March 2003.
- [7] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe. Design and implementation of a Routing Control Platform. In *ACM/USENIX NSDI*, 2005.
- [8] Matthew Caesar and Jennifer Rexford. BGP routing policies in ISP networks. *IEEE Network*, November 2005.
- [9] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Luang Lei. Quantifying Skype User Satisfaction. *ACM SIGCOMM*, September 2006.
- [10] Kuan-Ta Chen, Polly Huang, Guo-Shiuan Wang, Chun-Ying Huang, and Chin-Luang Lei. On the Sensitivity of Online Game Playing Time to Network QoS. *IEEE Infocom*, April 2006.
- [11] Cisco Systems. BGP Cost Community. Cisco IOS Documentation.
- [12] Nick Duffield, Kartik Gopalan, Michael R. Hines, Aman Shaikh, and Jacobus E. van der Merwe. Measurement Informed Route Selection. Passive and Active Measurement Conference, April 2007. Extended abstract.
- [13] Nick Feamster, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh, and Jacobus E. van der Merwe. The Case for Separating Routing from Routers. *ACM SIGCOMM FDNA*, Aug 2004.
- [14] Nick Feamster and Jennifer Rexford. A Model of BGP Routing for Network Engineering. In *SIGMETRICS*, June 2004.
- [15] Ramesh Govindan, Cengiz Alaettinoglu, Kannan Varadhan, and Deborah Estrin. Route servers for inter-domain routing. *J. Comp. Net. ISDN Sys.*, 30:1157–1174, 1998.
- [16] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A Clean Slate 4D Approach to Network Control and Management. *SIGCOMM CCR*, 35(5), 2005.
- [17] Timothy Griffin and Gordon T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *ICNP*, 2002.
- [18] Timothy G. Griffin and Gordon Wilfong. On the Correctness of IBGP Configuration. In *ACM SIGCOMM*, pages 17–29, New York, NY, USA, 2002. ACM Press.
- [19] D. Haskin. A BGP/IDRP Route Server alternative to a full mesh routing. IETF RFC 1863, October 1995.
- [20] Geoff Huston. Wither Routing? The ISP Column, November 2006.
- [21] Geoff Huston. Damping BGP. The ISP Column, June 2007.
- [22] Josh Karlin, Stephanie Forrest, and Jennifer Rexford. Pretty Good BGP: Improving BGP by Cautiously Adopting Routes. In *ICNP*, November 2006.
- [23] H. Khosravi and T. Anderson. Requirements for Separation of IP Control and Forwarding. IETF RFC 3654, November 2003.
- [24] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP Misconfiguration. *SIGCOMM Comput. Commun. Rev.*, 32(4):3–16, 2002.
- [25] Pedro Marques, Nischal Sheth, Robert Raszuk, Jared Mauch, and Danny McPherson. Dissemination of flow specification rules. Internet draft: draft-marques-idr-flow-spec-00.txt, June 2003.
- [26] D. McPherson, V. Gill, D. Walton, and A. Retana. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. RFC 3345, August 2002.
- [27] James Nichols and Mark Claypool. The Effects of Latency on Online Madden NFL Football. In *ACM NOSSDAV*, June 2004.
- [28] Nathan Patrick, Tom Scholl, Aman Shaikh, and Richard Steenberg. Peering Dagnet: Anti-Social Behavior Amongst Peers, and What You can Do About It. NANOG 38, October 2006.
- [29] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). IETF RFC 4271, 2006.
- [30] Alvaro Retana and Russ White. BGP Custom Decision Process. Internet draft: draft-retana-bgp-custom-decision-00.txt, October 2002.
- [31] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The End-to-End Effects of Internet Path Selection. In *ACM SIGCOMM*, September 1999.
- [32] Renata Teixeira, Timothy G. Griffin, Mauricio G. C. Resende, and Jennifer Rexford. TIE Breaking: Tunable Interdomain Egress Selection. In *CoNEXT*, 2005.
- [33] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *ACM SIGMETRICS*, 2004.
- [34] Jacobus E. van der Merwe et al. Dynamic Connectivity Management with an Intelligent Route Service Control Point. *ACM SIGCOMM INM*, October 2006.
- [35] Daniel Walton, Alvaro Retana, and Enke Chen. Advertisement of Multiple Paths in BGP. Internet draft: draft-walton-bgp-add-paths-05.txt, August 2006.
- [36] J. Wu, Z. Mao, J. Rexford, and J. Wang. Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *USENIX NSDI*, 2005.
- [37] Hong Yan, David A. Maltz, T. S. Eugene Ng, Hemant Gogineni, Hui Zhang, and Zheng Cai. Tesseract: A 4D Network Control Plane. In *ACM/USENIX NSDI*, 2007.