



Route servers for inter-domain routing ¹

Ramesh Govindan ^{*}, Cengiz Alaettinoğlu, Kannan Varadhan, Deborah Estrin

USC / Information Sciences Institute, 4676 Admiralty Way, 10th Floor, Marina del Rey, CA 90292-6695, USA

Abstract

Internet transmission and switching facilities are partitioned into different administrative *domains*. To effect routing between domains, domain *border routers* establish pairwise peering sessions and exchange routing information at *exchange points*. An alternative arrangement, in which each border router at an exchange point peers only with a *Route Server*, provides some operational benefits.

From a set-theoretic characterization of border router behavior, we derive a set-theoretic expression that completely and succinctly characterizes Route Server functionality. Performance measurements from our Route Server implementation reveal that the storage requirements of Route Servers can be much larger than that of a typical border router. We discuss a variety of techniques that can, in some cases, reduce Route Server storage requirements by a factor of five or more. © 1998 Elsevier Science B.V. All rights reserved.

1. Introduction

Internet resources, such as hosts, routers, and transmission facilities, are partitioned into different administrative *domains*. A campus or internal corporate network is an example of a domain. Data exchange between campus or corporate domains is facilitated by *provider domains*; these domains offer, as a service, transmission and switching facilities for data exchange between their customers' domains.

Further, by interconnecting with other providers, a provider may support data exchange between its customers and those of other providers.

To effect inter-domain data exchange, providers interconnect at *exchange points*. Typically, an exchange point is a high-speed subnetwork to which provider *border routers* (BRs) attach. At some exchange points, tens of BRs are expected to attach in the near future. MAE-East (where nearly twenty providers already attach), the Federal Internet Exchanges, and the London Internet Exchange, are examples of such exchange points. These existing exchange points have been established less from basic architectural considerations and more for economic reasons (e.g., operational cost amortization over providers connected to the exchange point); however, the National Science Foundation's proposed program for a commercial Internet [12] incorporates exchange points (called Network Access Points, or NAPs) into the routing infrastructure design, thereby conferring upon them "first-class"

^{*} Corresponding author. E-mail: govindan@isi.edu.

¹ This work was supported by the National Science Foundation under Cooperative Agreement NCR-9321043. The work of K. Varadhan and D. Estrin was supported by the National Science Foundation under contract number NCR-92-06418. Systems research at USC is supported through NSF infrastructure grant, award number CDA-9216321. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

status. With this development, and with the expected proliferation of Internet providers, the number of attached BRs at exchange points is likely to grow.

The routing requirements for data exchange between domains can be quite different from the routing requirements for data exchange involving the resources of a single domain. In the current Internet routing infrastructure, the former is supported by *inter-domain routing* and the latter by *intra-domain routing*. In the near future, inter-domain routing is likely to be achieved using the Inter-Domain Routing Protocol (IDRP²) [14]. IDRP supports data exchange between different domains, but in a manner that does not conflict with domain autonomy and heterogeneity. IDRP allows each domain to independently realize a wide variety of *policies*; broadly, such policies either express a domain's restrictions on access to its resources, or its preferences among other domains' resources for different classes of traffic.

In IDRP, domain BRs realize domain policy by restricting, based on configured *selection and export criteria*, the propagation of routing information. To exchange inter-domain routing information, a domain BR first establishes IDRP *peering sessions* (typically transport connections) with one or more other BRs. From these *peers*, the BR receives a set of IDRP *routes* – an IDRP route indicates its sender's reachability to an *address prefix* (a network-layer address aggregate). For each distinct address prefix, the BR chooses the route that best matches its selection criteria. Then, to each of its peers the BR propagates those selected routes that best match its export criteria. Section 2 formally characterizes this route selection and propagation behavior.

In theory, IDRP does not require global knowledge of domain routing policies. In practice, domain *policy databases* are proving essential for day-to-day Internet operations. Policy databases enable easier debugging of inter-domain routing anomalies, determination of Internet connectivity, and “what-if” analysis of routing policy. In North America, the Routing Arbiter project maintains the Routing Ar-

biter Data Base (RADB [4]); in Europe, RIPE [9] maintains a similar policy database.

Global knowledge of routing policies facilitates the use of *Route Servers* (RSs) at exchange points. At an exchange point, each provider BR usually establishes an IDRP peering session with every other BR; we call such a peering arrangement a *mesh* (Fig. 1). In *star* peering, instead, BRs at an exchange point peer only with an RS. The RS is configured, from a policy database, with the selection and export criteria of each BR at the exchange point. Each BR advertises its selected routes to the RS. Using this information, the RS performs IDRP route computation on behalf of the BRs. Star peering is *functionally equivalent* to mesh peering; that is, BRs at the exchange point select the same routes in either arrangement. In Section 3, we formulate the condition for this functional equivalence, and use it to formally derive RS behavior (Section 4).

Star peering has some operational advantages. First, it can simplify BR administration. In the mesh, domain administrators need to configure their domain BRs with route selection and export criteria. In the star, if domain BRs rely entirely on routes received from the RS, such configuration may be reduced. In the mesh, when a new BR attaches to the exchange point, every other BR must be configured to peer with this new BR; in the star, only the RS need be configured thus. Second, star peering reduces route processing at domain BRs – BRs may no longer need to apply selection and export criteria, since this is done on their behalf by RSs. As we show later, the star can reduce BR storage requirements as well. Anecdotal evidence suggests that some current high-end router products are limited in the number of peering sessions they can concurrently process [8]; memory limitations of Internet backbone routers are well documented [7]. Finally, since the RS is directly configured from the policy database, star peering at Internet exchange points is an incentive for maintaining the currency of registered policies.

Some of these advantages are particularly relevant at large exchange points. Further, at such exchange points, peering with the RS can be cost-effective for smaller Internet service providers. However, these advantages do not justify the RS as a fundamental component of the inter-domain routing architecture.

² The Border Gateway Protocol (BGP) version 4 [15] is currently deployed on the Internet. IDRP is a superset of BGP version 4. Our conclusions are equally applicable for BGP.

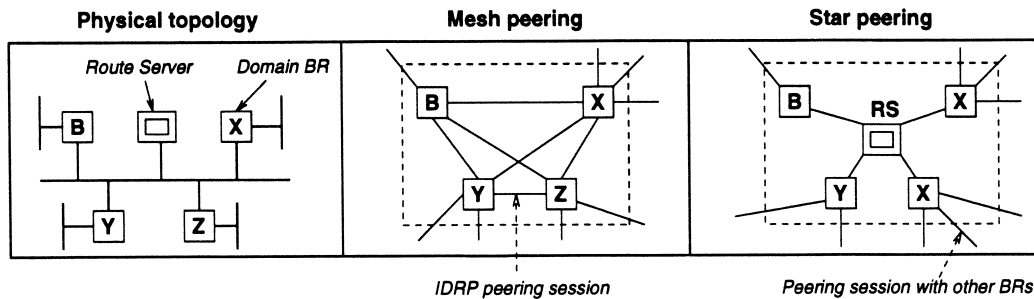


Fig. 1. Mesh peering. This figure shows four domain BRs attached to an exchange point, and the two possible logical IDRPeering topologies. In mesh peering, each BR peers with every other BR at the exchange point. In star peering, each BR peers only with an RS. In addition, a BR may also have peers not directly attached to the exchange point (e.g., other BRs in its own domain).

For instance, it is possible to design and implement mechanisms that automatically configure BRs directly from the policy database; such mechanisms would equally simplify administration and provide incentives for maintaining the currency of the policy database. Similarly, low-cost routers with greater processing power and increased memory capacity can reduce the need for RSs.

Star peering has two important drawbacks. First, an RS reduces BR processing and storage at the cost of considerably increasing its own processing and storage requirements (Section 5). Section 6 describes ways to reduce RS storage requirements. Second, an RS is a single point of failure at exchange points; backup RSs provide some failure tolerance, but add complexity to the peering arrangement (Section 7).

2. Set-theoretic characterization of IDRPeeking BRs

In Section 1, we briefly described how IDRPeeking BRs exchange routing information and perform route selection. In this section, we describe that process in greater detail, cast it in set-theoretic terms, and finally arrive at a set-theoretic expression for the BR's outputs in terms of its inputs.

2.1. Informal description of an IDRPeeking BR

Consider an IDRPeeking BR_a which peers with a number of BRs at one or more exchange points. BR_a may also peer with BRs in its own domain. From each of its peers, BR_a receives one or more IDRPeering routes. Each IDRPeering route contains an *address prefix*. An address prefix A represents the

topological region of the Internet in which hosts and routers' network-layer addresses have A as a prefix; a route from BR_a 's peer containing prefix A advertises that peer's reachability to some entity within, or in the vicinity of, that topological region. Address prefixes are usually represented by a network-layer address and a prefix length.

An IDRPeering route is associated with one or more *attributes*. Attributes convey information used by a route's recipients for packet forwarding or route selection. For example, the `NEXT_HOP` attribute contains the network-layer address that BR_a must use to forward packets destined for the address prefix indicated in the route; usually, the `NEXT_HOP` contains the network-layer address of the route's sender. Similarly, the value of the `RD_PATH` attribute is the list of identifiers of domains traversed by the route; the `RD_PATH` is used by BRs to avoid routing loops. When advertising a route, BR_a affixes its domain's identifier (all domains are assigned unique identifiers) to the route's `RD_PATH`.

We divide into four steps the procedure by which an IDRPeeking BR_a computes the routes that it advertises to its peers (Fig. 2):

- *Assigning route preferences*: From each of its peers, BR_a receives at most one route per address prefix³. For each of these routes, BR_a tests

³ This characterization is slightly inaccurate for IDRPeering. IDRPeering defines a QoS attribute; an IDRPeering BR may advertise two different routes to the same address prefix, but with different QoS values. For simplicity, we ignore this distinction in most of the rest of the paper. Section 2.3 briefly describes how our set-theoretic characterization may be modified to accommodate this.

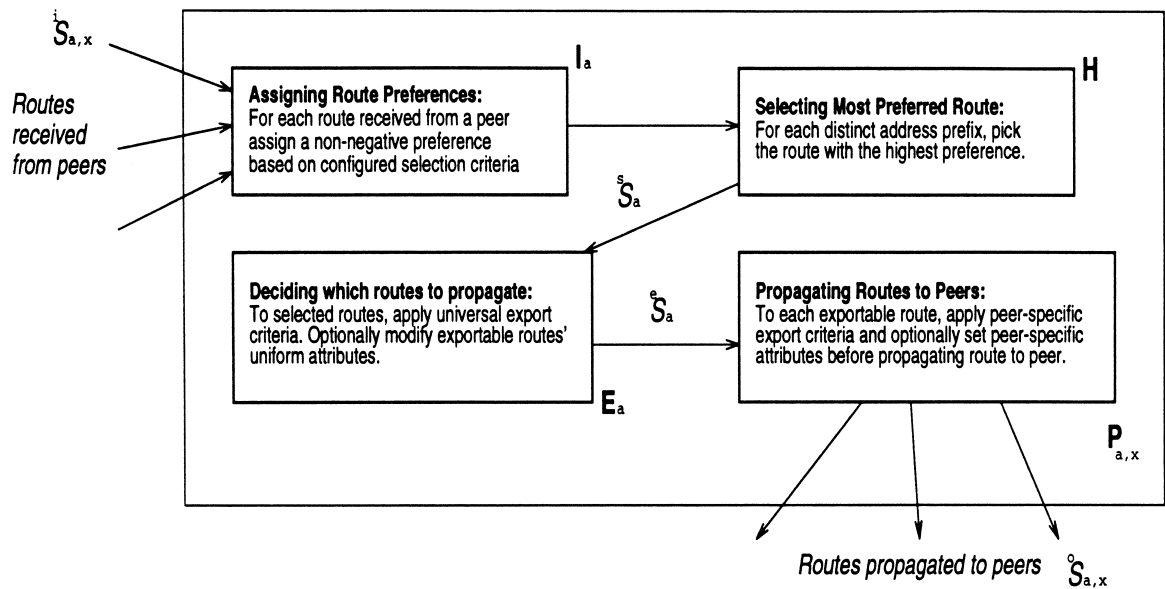


Fig. 2. BR Behavior: This figure describes the route selection and route propagation behavior of a IDRPspeaking BR. This pictorial depiction motivates a set-theoretic characterization of BR behavior.

whether its own domain identifier is present in the route's RD_PATH; if the test succeeds, the route is discarded. To each remaining route, the BR assigns either a non-negative integer *preference* or $-\infty$. The former indicates BR_a 's preference for that route, among all routes advertising reachability to the same address prefix; the latter indicates that BR_a will, under no circumstances, use the route to reach that address prefix.

Preference assignment decisions are based entirely on configured selection criteria [3]. Selection criteria are usually expressed in terms of the contents of a route – e.g., the values of its attributes. The statements “always assign a preference of 1 to routes heard from peer BR_x ” and “always assign $-\infty$ to routes originating from domain X” are examples of selection criteria. Selection criteria reflect BR_a 's preferences for the use of other domains' resources.

- *Selecting most preferred route:* In this step, BR_a selects, for each distinct address prefix, the route with the highest preference. If two or more routes have the same highest preference, IDRPs specifies tie-breaking rules based on the values of the routes' attributes.

- *Deciding which routes to propagate:* BR_a propagates some of its selected routes to each of its peers, based on configured export criteria. Export criteria express a domain's restrictions on access to its resources. These criteria fall into two categories: *universal* and *peer specific*. Universal export criteria determine the conditions under which a route may be propagated to a peer; the statement “do not export any route containing address prefix X to any peer” is an example of a universal export criterion. This step applies universal export criteria to each selected route; a route which fulfils these criteria is deemed *exportable*.

All route attributes fall into one of two categories; one, in which all route recipients see the same value for the attribute (*uniform* attributes), and another in which different recipients may see different values of the attribute (*peer-specific* attributes) for the same selected route. In this step, BR_a may assign values to uniform attributes. The EXT_INFO is an example of a uniform attribute; it contains information about the origin of the route advertisement.

- *Propagating routes to peers:* In this final step,

BR_a decides which of the exportable routes to actually propagate to each of its peers. To do this, it applies configured peer-specific export criteria to each exportable route. The statement “do not export routes with domain Y in their RD_PATH to peer BR_x ” is an example of a peer-specific export criterion for peer BR_x . Only exportable routes that fulfil peer-specific export criteria for peer BR_x are propagated to that peer.

Before propagating a route, BR_a may set the route’s peer-specific attributes. The value to assign to the peer-specific attribute is usually determined from configured information. An example of a peer-specific attribute is the `MULTI_EXIT_DISC`. If BR_a has two or more exit points to a neighboring domain, the value assigned to this attribute can be used to discriminate between which of these exit points the neighboring domain must use to forward packets. BR_a may advertise different values of this attribute to different peers.

2.2. Set-theoretic characterization

In this section, we formally characterize the four-step procedure described in Section 2.1. Specifically, we arrive at a set-theoretic description of an IDRP-speaking BR’s output in terms of its inputs.

We denote an IDRP route by the pair \langle address prefix, route attributes \rangle . In general, *route attributes* themselves may be described as a sequence of \langle attribute, value \rangle pairs. We ignore this detail for notational simplicity.

The inputs to the four-step procedure of Section 2.1, and the results of three of the four steps can be represented by *route sets* (see Fig. 2). A route set is a set of pairs denoting IDRP routes, with at most one route pair for every distinct address prefix. For an IDRP-speaking BR_a , we denote these route sets by:

- ${}^iS_{a,x}$, the inputs to BR_a , are the route sets that peers BR_x advertise to BR_a .
- sS_a is the route set selected by BR_a , from among received routes that match its selection criteria.
- eS_a is the route set representing BR_a ’s exportable routes.
- ${}^oS_{a,x}$, BR_a ’s outputs, are the route sets that BR_a propagates to each of its peers BR_x . By our notation, ${}^iS_{a,x} \equiv {}^oS_{x,a}$.

The first two steps in the four-step procedure (Section 2.1) assign route preferences based on selection criteria (sometimes called **import** criteria) and select the **highest** preference route. If we denote the computation performed in these two steps respectively by set functions I_a and H (Fig. 2), we can write the following equation for the output of the second step:

$${}^sS_a = H\left(\bigcup_x (I_a({}^iS_{a,x}))\right), \quad (1)$$

where, for each route r in its argument set, I_a computes its result set using the following procedure:

1. Test whether r ’s RD_PATH already contains BR_a ’s domain identifier. If the test fails (r does not have a looped path vector), proceed to the next step.
2. From the configured selection criteria, assign a preference value to route r . Insert this \langle route, preference \rangle pair into the result set. *Preference* is either a non-negative integer, or $-\infty$.

The function H takes a set of \langle route, preference \rangle pairs as its argument and computes its result set using the following procedure:

1. Partition the input set into subsets, such that in each subset, the address prefixes of the route elements of every \langle route, preference \rangle pair are identical.
2. From each subset formed in the previous step, select the route associated with the highest preference for inclusion in the result set. If all routes in the subset are assigned $-\infty$, do not select any route for inclusion in the result set. If more than one route in the subset has the highest preference, break ties according to rules specified in [3].

The third step in Section 2.1 applies BR_a ’s universal export criteria to compute its exportable routes. If we represent the computation performed in this step by the function E_a (Fig. 2), we can write the following equation for the output of the third step:

$${}^eS_a = E_a({}^sS_a). \quad (2)$$

For each route r in its argument set, E_a computes its result set using the following procedure:

1. Test whether BR_a ’s configured universal export criteria indicate that r can be propagated to BR_a ’s peers. If the test succeeds, proceed onto the next step.

2. Set each uniform attribute in r appropriately. Include the resulting route in the result set. For example, this step would set r 's EXT_INFO attribute if BR_a were originating the advertisement for route r . A more detailed description of an IDRPspeaker's handling of such attributes may be found in [3].

Finally, the fourth step described in Section 2.1 applies BR_a 's peer-specific export criteria to compute BR_a 's outputs. If we represent the computation performed in this step by the function $P_{a,y}$, we can write the following equation for BR_a 's output to peer BR_y :

$${}^o S_{a,y} = P_{a,y}({}^e S_a). \quad (3)$$

$P_{a,y}$ is a function that maps its argument set (a set of routes) into another route set; for each r in its argument set, $P_{a,y}$ computes its result set using the following procedure:

1. Test whether BR_a 's configured peer-specific export criteria indicate that r can be propagated to BR_y . If the test succeeds, proceed to the next step.
2. Set each peer-specific attribute in r appropriately. Include the resulting route in the result set. For example, this step would either set a configured value for the MULTI_EXIT_DISC attribute or pass unmodified the received value (which may be, for instance, a metric learned from intra-domain routing). A more detailed description of an IDRPspeaker's handling of such attributes may be found in Ref. [3].

Combining Eqs. (1)–(3), we can easily express BR_a 's outputs – the sets ${}^o S_{a,y}$ – in terms of its inputs, the sets ${}^i S_{a,x}$.

2.3. Discussion

For brevity, we have just sketched the definitions of set functions I , H , P , and E . That BR behavior can be characterized thus is, in and of itself, neither surprising nor interesting; the behavior of any another deterministic computation can probably be described in a similar manner. However, as we show in Section 4, this characterization enables us to make formal assertions about RS behavior.

This set-theoretic characterization succinctly captures BR route selection and route propagation be-

havior. However, it is not a minimal characterization; it is possible, for instance, to fold the functions E_a and $P_{a,x}$ into one set function. This would result in an expression for the ${}^o S_{a,x}$ sets directly in terms of ${}^i S_a$. Our characterization simplifies the derivation of RS behavior (Section 4.1).

Our use of the terms ‘‘input’’ and ‘‘output’’ route sets do not imply that IDRPspeaking BRs periodically recompute and advertise routes to their peers; on the contrary, IDRPspeaking BRs usually send a route advertisement only when a route to some address prefix changes. Eqs. (1)–(3) should be viewed as an invariant satisfied by each IDRPspeaking BR *at every instant*.

A domain BR may choose to propagate routes learned from intra-domain routing; this aspect of BR behavior is not expressed in our set-theoretic characterization, which models only IDRPspeaking. It is possible to trivially extend our characterization to represent this explicitly, for example by modeling intra-domain route sets as being learned from an ‘‘intra-domain peer’’.

We have assumed that an IDRPspeaking BR advertises to its peer at most one route per address prefix. Actually, an IDRPspeaking BR may advertise, at most one route per address prefix per QoS value (or the values of other IDRPspeaking attributes like EXPENSE and TRANSIT_DELAY [3]). Our characterization can be easily extended to model this behavior, for example by distinguishing each route set defined in Section 2.2 by different QoS values.

3. Functional equivalence between mesh and star peering

In this section, we define the condition for functional equivalence between *complete mesh* and *complete star* peering at exchange points. In a complete mesh at exchange point \mathcal{L} , every BR attached to \mathcal{L} peers with every other BR attached to \mathcal{L} . In a ‘‘corresponding’’ complete star at \mathcal{L} , every BR attached to \mathcal{L} has the same selection and export criteria as in the complete mesh, and peers only with the RS. In this and the next section, we use the terms mesh and star to respectively mean (unless otherwise specified), a complete mesh and a complete star.

Later we show that the RS behavior we derive from this equivalence condition can be used even when mixed mesh and star are deployed at \mathcal{L} .

We say that the mesh and its corresponding star are functionally equivalent if, given the same inputs, all BRs at the exchange point select the same routes in both arrangements. Thus, if in Fig. 1, the routes advertised over the IDRPs peering “links” into the “box” are the same in both arrangements, the routes selected by each BR within the “box” must also be the same. Further, this condition must hold for all possible selection and export criteria at those BRs. If our equivalence condition holds, the routes advertised over the IDRPs peering “links” out of the “box” are the same in both arrangements (from Eqs. (2) and (3)).

Our equivalence condition can be stated in terms of the notation developed in Section 2. Denote by ${}^{\#s}S_a$ the set sS_a when BRs at \mathcal{L} are configured in the mesh. Denote by ${}^{\star s}S_a$ the set sS_a when BRs at \mathcal{L} are configured in the corresponding star. Using a similar notation for the sets ${}^iS_{a,x}$ we get:

Equivalence condition. For each BR_a in \mathcal{L} , and for all possible selection and export criteria at BR_a , if ${}^{\#i}S_{a,g} = {}^{\star i}S_{a,g}$ for all peers BR_g of BR_a which are not attached to \mathcal{L} , we require

$${}^{\#s}S_a = {}^{\star s}S_a. \quad (4)$$

4. A set-theoretic characterization of route servers

Using the formalism developed in Section 2, and starting from the equivalence condition of the previous section, we can mathematically *derive* an expression for the Route Server’s outputs in terms of its inputs. This section presents that derivation, and the interpretation of the derived expression.

4.1. Derivation of route server behavior

In a star, the RS peers with each BR_a attached to \mathcal{L} . The RS receives a set of routes from each BR_a and advertises another set of routes to each BR_a . Denote the former route sets by iR_a (the RS’s inputs) and the latter by oR_a (the RS’s outputs). BR_a ’s selection and export criteria in the star are the same

as in the mesh; that is, the functions I_a , E_a , and $P_{a,x}$ are the same in both arrangements. The star, however, introduces a new peer-specific export function $P_{a,RS}$. If the star is functionally equivalent to the mesh, we show that there exists a set-theoretic expression (Eq. (11)) for oR_a in terms of iR_a and the functions I and P .

By re-writing Eq. (1), we arrive at the following expressions for ${}^{\#s}S_a$ and ${}^{\star s}S_a$:

$${}^{\#s}S_a = H \left(\left(\bigcup_{x, BR_x \text{ not on } \mathcal{L}} I_a({}^{\#i}S_{a,x}) \right) \cup \left(\bigcup_{x, BR_x \text{ on } \mathcal{L}} I_a({}^{\#i}S_{a,x}) \right) \right), \quad (5)$$

$${}^{\star s}S_a = H \left(\left(\bigcup_{x, BR_x \text{ not on } \mathcal{L}} I_a({}^{\star i}S_{a,x}) \right) \cup \left(I_a({}^oR_a) \right) \right). \quad (6)$$

Both equations simply split the set union of Eq. (1) into two parts: the route sets advertised by peers not directly attached to \mathcal{L} , and the route sets advertised by peers directly attached to \mathcal{L} .

The contribution to ${}^{\#s}S_a$ in Eq. (5) from peers directly attached to \mathcal{L} can be rewritten as the set of $\langle \text{route, preference} \rangle$ pairs having the highest preference among routes advertised by those peers. Eq. (5) then becomes:

$${}^{\#s}S_a = H \left(\left(\bigcup_{x, BR_x \text{ not on } \mathcal{L}} I_a({}^{\#i}S_{a,x}) \right) \cup \left(I_a \left(H \left(\bigcup_{x, BR_x \text{ is on } \mathcal{L}} I_a({}^{\#i}S_{a,x}) \right) \right) \right) \right). \quad (7)$$

Given the same inputs to BR_a , in both arrangements, from peers not attached to \mathcal{L} , we see by an inspection of Eqs. (6) and (7) that:

$${}^oR_a = H \left(\bigcup_{x, BR_x \text{ on } \mathcal{L}} I_a({}^{\#i}S_{a,x}) \right). \quad (8)$$

By noticing that ${}^iS_{a,x}$ can be written as ${}^oS_{x,a}$, and the latter set can be re-written in terms of eS_x (Eq. (3)), we get the following expression:

$${}^oR_a = H \left(\bigcup_{x, BR_x \text{ on } \mathcal{L}} I_a \left(P_{x,a}({}^{\#e}S_x) \right) \right). \quad (9)$$

Recall that ${}^{\#e}S_x$ can be written in terms of ${}^{\#s}S_x$ (Eq. (2)); and, if the star is equivalent to the mesh, then we can use Eq. (4) to re-write Eq. (9) as:

$${}^oR_a = H \left(\bigcup_{x, BR_x \text{ on } \mathcal{L}} I_a \left(P_{x,a}({}^{\star e}S_x) \right) \right). \quad (10)$$

Finally, if $P_{a,RS}$ is the identity function, then ${}^{\star e}S_x$ is simply the set of routes that BR_a actually propa-

gates to the RS in the star (from Eq. (3)); we have called this set iR_x . Making this substitution, we get

$${}^oR_a = H \left(\bigcup_{x, BR_x \text{ on } \mathcal{R}} I_a \left(P_{x,a} \left({}^iR_x \right) \right) \right). \quad (11)$$

To summarize, when the equivalence condition of Section 3 holds, the RS must compute and advertise for each BR_a , the set oR_a from Eq. (11); further, the set of routes that each BR_a propagates to the RS is defined by the requirement that $P_{a,RS}$ be the identity function. For the purposes of this paper, Eq. (11) suffices as a succinct description of RS behavior in the star. For completeness, we must prove the converse – we must show that given the same set of inputs to the mesh and star and RS behavior defined by Eq. (11), each BR selects the same routes in both arrangements. We refer the interested reader to Appendix A for this.

4.2. Informal description of route servers

Eq. (11) completely and succinctly defines the behavior of a RS in the star. Based on this equation, we may describe the RS's computation of oR_a as a three-step procedure (Fig. 3). This procedure is repeated for every BR that peers with the RS.

- *Applying peer-specific export criteria:* In this step, to routes received from each peer BR_x , the Route

Server applies the *same* peer-specific export criteria that BR_x would apply before propagating routes to BR_a . Of those routes that match this criteria, the RS sets peer-specific attributes *in the same way* that BR_x would have before propagating routes to BR_a .

For this, the RS must be configured with each client BR_a 's peer-specific export criteria and peer-specific attribute handling rules. Further, for the RS to correctly emulate a BR's peer-specific attribute handling behavior, each BR must pass on a route's peer-specific attributes unchanged to the RS (this is actually implied by the condition that $P_{a,RS}$ be the identity function).

- *Assigning route preferences:* To each route resulting from the previous step, the RS assigns *the same preference* that BR_a would assign the route. Again, in order to correctly emulate BR_a , the RS must be configured with BR_a 's selection criteria. In this step, the RS is able to correctly apply BR_a 's selection criteria because it has correctly set the appropriate peer-specific attributes *and* because BR_x has correctly applied uniform attributes before advertising routes to the RS. The RS itself does not set any uniform attributes.
- *Selecting most preferred route:* From the routes and their associated preferences obtained from the

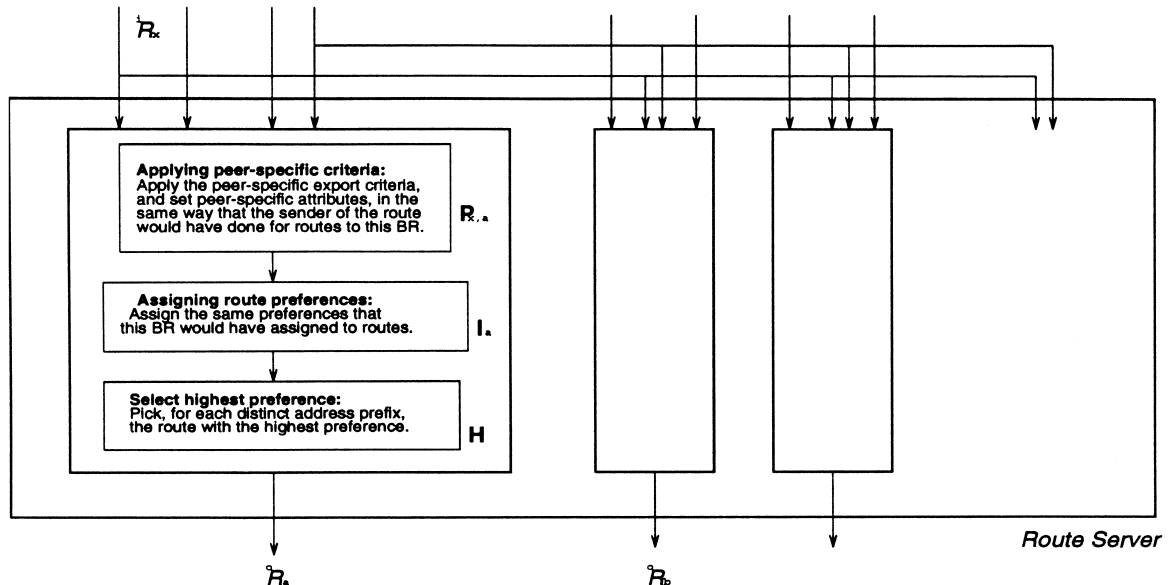


Fig. 3. RS Route selection: This figure pictorially describes the route selection behavior of Eq. (11).

previous step, the RS selects, for each distinct address prefix, the most preferred route. If all routes to an address prefix are assigned $-\infty$, the RS does not select any routes. The set of routes selected in this step forms oR_a .

4.3. Discussion

The equivalence condition of Section 3 was stated in terms of complete mesh and complete star peering. However, if BR_a peers with the RS as well as with some BR_y attached to \mathcal{L} , that arrangement is still functionally equivalent to complete mesh peering; if r is a route in ${}^{\#s}S_a$ received from BR_y in the complete mesh, BR_a receives two copies of r in this new arrangement – one from the RS and one directly from BR_y . The more general arrangement, in which every BR_x at \mathcal{L} peers with every other BR_y at \mathcal{L} either directly, or indirectly through the RS (that is, both BR_x and BR_y peer with the RS), or both (redundantly), is also functionally equivalent to the complete mesh.

Eq. (11) shows that star peering can reduce route processing and route storage at BRs. In mesh peering, to routes received from or advertised to each BR_x on \mathcal{L} , BR_a applies selection and export criteria; in star peering, the RS performs this computation. In mesh peering, BR_a may receive one route to an address prefix d from each BR_x at \mathcal{L} ; in star peering, BR_a receives at most one route to d from the RS (a BR must store each received advertisement, since IDRPs speakers only advertise updates).

Our derivation has implicitly assumed that, to a BR in a star, the RS appears to be an IDRPs speaker. Both Eq. (11) and the informal description of the previous section show, however, that the RS's route selection procedure is different from that of an IDRPs speaker. In particular, the characterization described in Section 2 does not apply to the RS. There is, however, significant benefit to using IDRPs peer connection management and route exchange mechanisms for peering between RS and the BRs in the star. This avoids inventing a completely new route exchange mechanism. Further, no new code need be deployed at BRs to peer with the RS.

As with the characterization of BR behavior, our description of RS behavior does not imply that the RS periodically computes and propagates the routes

in oR_a . Whenever some peer advertises a new route to an address prefix, the RS updates oR_a for each BR_a and advertises only updates. Eq. (11) is an invariant satisfied by the RS, for each BR_a , at all times.

In the star, even though routing information between two BRs attached to \mathcal{L} passes through the RS, it is desirable that packet forwarding between those two BRs bypass the RS completely. This is achieved using the NEXT_HOP attribute; if oR_a contains a route r from BR_x , r 's NEXT_HOP attribute contains BR_x 's network-layer address. Using this address, BR_a forwards data packets directly to BR_x .

In IDRPs, a BR is not required to assign a value to the NEXT_HOP attribute of a route it propagates [3]; recipients of such a route implicitly assume that the value of this attribute is the sender's network-layer address. Therefore, in the mesh, BR_a may receive the same route r from BR_x , but without the NEXT_HOP set; in the corresponding star, though, this attribute would be set. This apparent violation of our equivalence characterization is readily solved by explicitly considering r to be associated with a NEXT_HOP attribute in the mesh, even if BR_x did not assign a value to that attribute before propagating r to BR_a .

5. Route server performance

The RS described in Section 4.1 can be implemented in different ways. A potentially space efficient implementation maintains a single table containing all inputs to the RS, i.e., iR_a for all BR_a at \mathcal{L} . Suppose BR_x advertises a new route to address prefix d . To compute the effect of this change on oR_a , this implementation must apply $P_{x,a}$ and I_a to all routes to d in its table. It must repeat this computation for every BR_a . Such an implementation strategy is computationally intensive and could adversely affect global route convergence times.

Our initial RS implementation makes a different time/space trade-off. On behalf of each BR, our RS implementation conceptually maintains a table, called a *view*, (the view for BR_a is denoted by $view_a$), which stores the result of

$$\bigcup_{x, BR_x \text{ on } \mathcal{L}} I_a \left(P_{x,a} \left({}^iR_x \right) \right)$$

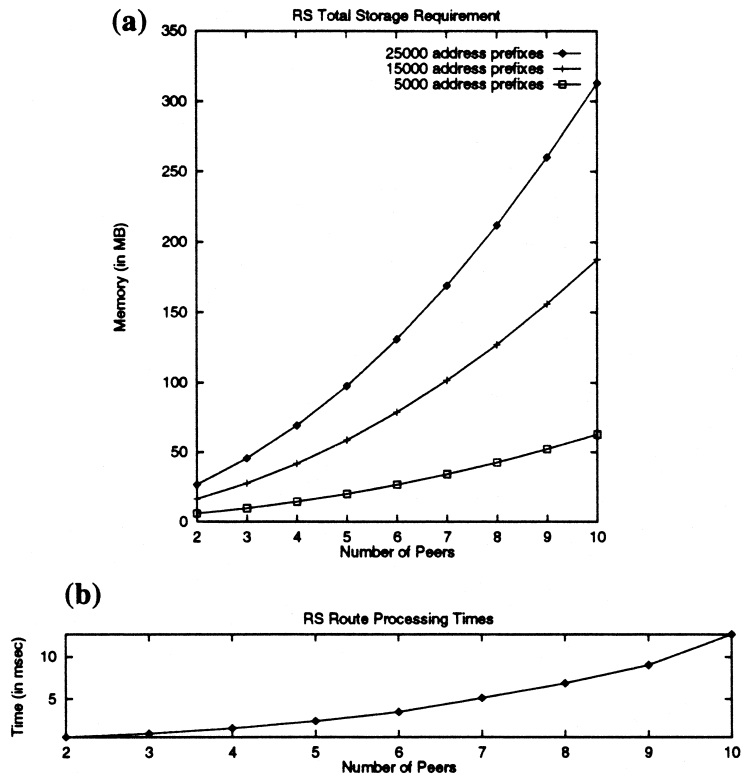


Fig. 4. RS performance: (a) shows the variation of the RS's total storage requirement with the number of peers. Even with 10 peers, a significant amount of memory (310 MB) is required for route storage. (b) shows the variation of RS route processing time with the number of peers. Only 12 ms are required to process a topology change affecting one address prefix, for ten peers.

(see Eq. (11)). More precisely, $view_a$ contains a row for each address prefix and a column for each peer. Let r be the route to address prefix d that is received by the RS from peer BR_x . The entry in the row for d and column for BR_x contains the \langle route, preference \rangle pair formed by first applying $P_{x,a}$ to r , and then applying I_a to the resulting route. When the RS receives a new route to d from some BR_x , it can efficiently compute the effect of this change on oR_a from $view_a$.

Unlike our RS, an IDRP-speaking BR maintains only one "view" – its own. Therefore, our RS's *total storage requirement* is likely to be significantly larger than that of a BR. If a route to an address prefix changes, the BR need only compute the effect of this route on its "view". However, our RS must compute the effect of this change on every view; the time to do this (*route processing time*) is also likely to be larger for the RS than a BR.

For our RS implementation, we can easily analyze the asymptotic behavior of these metrics. Consider an exchange point \mathcal{L} at which N BRs peer with a RS. If, in the worst case, each BR advertises routes to m address prefixes, the RS's total storage requirement is $O(mN^2)$ – each view requires $O(mN)$ storage (from Eq. (11)). Further, the route processing time is $O(N^2)$ – a topology change affecting a single address prefix can result in $O(N)$ processing per view. For an IDRP-speaking BR (from Eq. (1)), the storage requirement is $O(mN)$ and the route processing time is $O(N)$.

We also instrumented our RS implementation to measure⁴ the actual values of these metrics for some synthetic workloads. In our experiments, the

⁴ Our RS implementation is for BGP version 4. We do not expect our conclusions to differ for an IDRP RS.

RS ran on a SPARCStation 20/41. In each “run” of the experiment, the RS opened a BGP peering session with N other “border routers” (SPARCStation 10s and 20s), received routes to m address prefixes from each BR, and computed and advertised each BR’s view. We instrumented the RS to measure, after each run, the RS’s total storage requirement and route processing time. The former was measured as the maximum size of the RS process’ data segment during the run; the latter as the RS process’ CPU utilization between receiving the first route and sending the last route, divided by m . We varied m from 1000 to 25 000 (in increments of 1000) and N from 2 to 10.

Fig. 4 shows the RS’s route processing time when m is 25 000. Only 12 ms are required to process a topology change affecting one address prefix, for ten peers. The curve fits a parabola in keeping with our analysis above. The parabola is quite “shallow” upto 15 peers; that is, for upto about 15 peers our RS will process typical topology changes – those affecting tens of address prefixes – in about a second. However, the RS can take on the order of minutes to compute views after catastrophic events, e.g., a link failure at the exchange point which causes all peer connections fail and simultaneously restart.

Fig. 4 also shows the RS’s total storage requirement, for three different values of m . These curves also fit a parabola, validating our analysis above. The storage requirement for 10 peers and 25 000 address prefixes is more than 300 MB; extrapolating the 25 000 address prefix curve to 15 peers, we arrive at a storage requirement of nearly 650 MB. Backbone routers on the Internet already carry more than 25 000 address prefixes; it is unrealistic to expect workstation memory capacity to keep pace with a RS with tens of peers each of whom might advertise routes to a significant fraction of these address prefixes.

Therefore, while the RS route processing times are considerable⁵, workstation memory capacity is

likely to be the initial barrier limiting the use of RSs at large exchange points.

6. View merging and decomposition

In this section, we present sufficient conditions under which two or more views can be *merged*, or decomposed into *subviews*, to reduce RS storage and computation requirements. View decomposition, in particular, can reduce RS storage requirements significantly; we evaluate view decomposition performance under different scenarios.

6.1. Merging and decomposition

If, for BRs BR_x and BR_y , oR_x and oR_y evaluate to the same set for the same inputs, the RS need only maintain a single view on behalf of the two BRs. Intuitively, $view_x$ and $view_y$ can be “merged” when both BRs’ selection criteria are identical *and* all other peers’ peer-specific export criteria are the same for BR_x and BR_y . Formally, the condition for merging the two views is, by inspection of Eq. (11):

Merge condition 1.

$$\forall a, BR_a \text{ on } \mathcal{L} : I_x \circ P_{a,x} \equiv I_y \circ P_{a,y} \text{ }^6.$$

To motivate the decomposition of views into subviews, we describe a different representation of merge condition 1. Recall that $view_x$ is conceptually a table with a row for each address prefix d and a column for each BR_a at \mathcal{L} (Section 5). The entry in row d and column a of $view_x$ is calculated by the following function:

$$entry_{x,d,a}(r) = \begin{cases} I_x \circ P_{a,x}(r) & r \text{ is a route to address-prefix } d \\ & \text{received from peer } BR_a, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

That is, to routes containing address prefix d from BR_a , this *entry* function applies BR_a ’s peer-specific

⁵ The route processing times reported here do not capture the processing of complex selection criteria, such as preference assignment based on path vector regular expressions. While such criteria are likely to be expressed in a future commercial Internet, current Internet domain selection criteria remain relatively simple.

⁶ Two functions are equivalent if they have the same domain and for any value in the domain both functions map to the same value.

export criterion for BR_x and then applies BR_x 's route selection criteria. Using this, we can define a row function that computes row d of $view_x$:

$$row_{x,d}(r_1, \dots, r_N) = \langle entry_{x,d,p_1}(r_1), \dots, entry_{x,d,p_N}(r_N) \rangle, \quad (12)$$

where N is the number of peers. This row function represents the processing performed by the RS, on behalf of BR_x , on routes containing address prefix d . Using this notation, merge condition 1 can be re-written as:

Merge condition 1.

$$\forall d: row_{x,d} \equiv row_{y,d}.$$

RS processing is determined by BR selection and export criteria at \mathcal{R} ; for merge condition 1 to be satisfied, these criteria must be such that the RS identically processes routes to every address prefix on behalf of both BR_x and BR_y . An analysis of current domain policies [16,13] reveals that the likelihood of this event is very small; further, this situation is unlikely to change in a future commercial Internet with even more diverse domain selection and export criteria. It is much more likely that RS processing on behalf of two BRs will be identical for some subset of address prefixes. That is, it is more

likely that $view_x$ and $view_y$ have some common row functions. When this is true, we can decompose the two views into three subviews. These subviews respectively contain (Fig. 5): (1) the rows with common row functions, (2) the rows of $view_x$ whose row functions are different from the corresponding row functions of $view_y$, and (3) the rows of $view_y$ whose row functions are different from the corresponding row functions of $view_x$. Formally, we can decompose $view_x$ and $view_y$ into three subviews if, for a subset of address prefixes D ,

Merge condition 2.

$$\forall d \in D: row_{x,d} \equiv row_{y,d} \wedge \forall d \notin D: row_{x,d} \neq row_{y,d}.$$

When two views are decomposed into three subviews, the RS need only keep one copy of the common subview; without this decomposition, it would have required additional storage for the common subview. Greater savings may be obtained by repeatedly applying merge condition 2 to the collection of N views (one for each BR). The following procedure achieves maximum space savings from view decomposition. Starting from the collection of N views (one for each BR), the procedure selects any two views in the collection, applies merge condi-

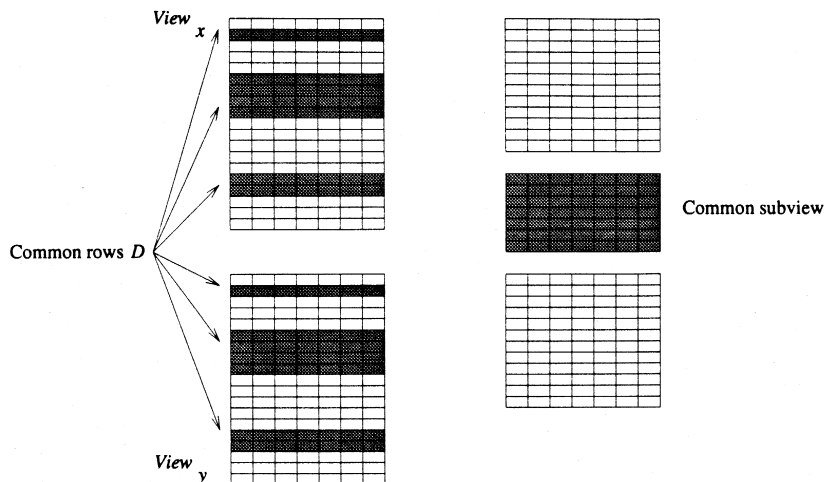


Fig. 5. View decomposition: Views for peers BR_x and BR_y may be decomposed into three subviews: one containing the common rows in $view_x$ and $view_y$, the other two containing the remnants from the two views.

tion 2 to those views, and replaces the two selected views with the resulting three subviews in the collection. This step is repeated until no further decomposition is possible. This procedure has exponential complexity; a discussion of heuristics for efficiently computing subviews is beyond the scope of this paper. As a result of this procedure, a route to address prefix d can appear in as many subviews as there are distinct *row* functions for d ; in the worst case, there may be N distinct *row* functions for d .

6.2. Analysis and evaluation of decomposition

In the current Internet, domain selection and export criteria are expressed using routing policy specification languages described in [10,2]. While, in general, *row* functions can be specified in terms of any route attribute, these languages allow *row* functions to only depend on a route's sender and on the address prefix associated with the route⁷. In this section, we first derive the number of possible *row* functions for address prefix d under this restriction. Then, assuming all *row* functions are equally likely, we derive the probability of N BRs having exactly v distinct *row* functions. We use this probability to estimate RS storage requirements with view decomposition, and compare it to RS storage requirement without view decomposition.

Using a simple combinatorial argument, we count the number of possible *row* functions when a route to address prefix d is advertised by p BRs. Intuitively, this is the number of possible choices for selection and export criteria for p different routes to d ; a BR may “reject” some of these p routes, then assign a rank order (the preference assignment step in Section 2.1) to the remaining routes. Thus, if BR_a rejects some i routes, it can assign one of $(p-i)$ preference values to the remaining routes; we can select i routes for rejection in $\binom{p}{i}$ ways, and assign $(p-i)$ preference values to the rest in $(p-i)!$

ways⁸. Then, the total number of *row* functions that a BR_a can choose from for address prefix d is given by:

$$\rho(p) = \sum_{i=0}^p (p-i)! \times \binom{p}{i}.$$

Since there are N BRs and each BR may choose a *row* function independently, the total number of combinations of *row* functions equals $\rho(p)^N$. Among these, $\binom{\rho(p)}{v} \times v! \times v^{N-v}$ have exactly v distinct *row* functions; v distinct *row* functions can be chosen out of $\rho(p)$ functions in $\binom{\rho(p)}{v}$ ways, these v functions can be arranged among v BRs in $v!$ ways, and the remaining $(N-v)$ BRs can be assigned these v functions in v^{N-v} ways. If we assume that all *row* functions are equally likely, the probability that there are exactly v distinct *row* functions for address prefix d among N BRs is given by

$$\mathcal{P}(v) = \frac{\binom{\rho(p)}{v} \times v! \times v^{N-v}}{\rho(p)^N}.$$

The expected number of distinct *row* functions for d (or, equivalently, the expected number of subviews that a route to d is installed in), when p BRs advertise a route to d , is given by $\sum_{v=0}^N v \times \mathcal{P}(v)$. Thus, when merge condition 2 is applied to *row* functions for d , the expected RS storage requirement per address prefix is proportional to $p \times \sum_{v=0}^N v \times \mathcal{P}(v)$. In the worst case, all the p routes to d are installed in each of N subviews; the RS's storage requirement for routes to d is then proportional to $p \times N$.

6.3. Discussion

Fig. 6a plots RS storage requirements using view decomposition to RS storage requirement without view decomposition, as a function of p , for different values of N . From this figure, we see that significant savings are possible for small values of p . For example, in an exchange point with 20 BRs (the size

⁷ Actually, RIPE-181, a policy specification language [2], allows *row* functions to depend on the MULTI_EXIT_DISC (Section 2.1) attribute as well. Since the likelihood of more than one BR from the same domain at one exchange point is very low, we do not consider the attribute handling functions that use MULTI_EXIT_DISC in our analysis.

⁸ $(p-i)!$ does not include rank ordering where two peers have the same rank for routes to d . This is intentional, since the IDR tie breaking rules actually achieve total ordering.

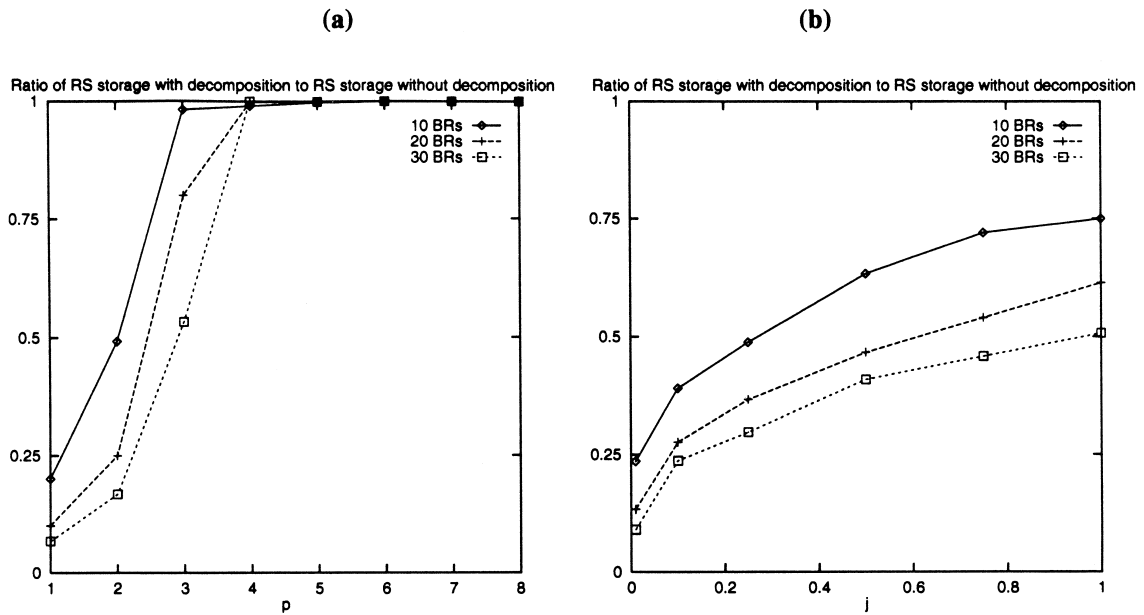


Fig. 6. Performance of view decomposition: The abscissa of graph (a) is p , the number of BRs advertising routes to a given address prefix d . The ordinate shows the ratio of the expected RS storage requirement to the worst case RS storage requirement, for different values of p . The abscissa of graph (b) is j , the fraction of possible *row* functions that are actually used by BRs. The ordinate of this graph is the ratio of the expected RS storage requirement to the worst case RS storage requirement when p is distributed as shown in Table 1.

of some existing exchange points), view decomposition can reduce RS storage requirements by a factor of 5. For larger p , view decomposition does not yield much benefit; the number of *row* functions for large p is large enough (e.g., 65 *row* functions for $p=4$) that the probability of two BRs choosing the same *row* function is very low. As expected, the benefits of view decomposition are higher for larger N .

In Fig. 6a, we vary p from 1 to N – in practice, not all values of p are equally likely. Preliminary data from 12 BRs at the MAE-East exchange point shows a fairly skewed distribution for p (Table 1). For this distribution, view decomposition reduces RS

route storage requirements by more than a factor of 5. We believe that, in the future, the distribution for p is likely to be less skewed. An approximation of such a distribution is obtained by modeling the old NSFNET backbone (with more than 100 peers) as an exchange point (Table 1).

An examination of Fig. 6a shows that view decomposition does not provide more than a factor of two savings, for this distribution of p . However, that analysis assumes that each BR chooses its *row* functions uniformly among $\rho(p)$ *row* functions; in practice, we expect this distribution to be skewed towards a subset of “common” *row* functions. Let j be that fraction of the $\rho(p)$ possible *row* functions

Table 1

Distribution for p : this table shows the fraction of address prefixes to which routes are advertised by p BRs on the NSFNET backbone, for different values of p

| p | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|--------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| MAE-East | 0.9424 | 0.0576 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NSF Backbone | 0.11376 | 0.29845 | 0.29467 | 0.13522 | 0.12860 | 0.02485 | 0.00336 | 0.00098 | 0.00006 |

that are actually considered for computing $\mathcal{P}(v)$. Fig. 6b shows the ratio of the RS storage requirements with view decomposition to RS storage requirements without view decomposition, as a function of j , for the NSFNET's distribution of p . This figure indicates significant savings for low values of j , as expected. For an exchange point with 20 BRs, view decomposition reduces the storage requirement by a factor of 7 for $j = 0.01$, and by a factor of 3 for $j = 0.1$ ⁹.

7. Multiple route servers

In Section 4.3, we described the partial star/partial mesh alternative to the complete star. This section discusses a variant of the star in which more than one RS is employed at an exchange point. Such an arrangement has many uses, including improving the failure tolerance of the complete star, and reducing RS storage and computation requirements.

7.1. Robustness

In the complete star, if each BR only peers with the RS, the latter becomes a single point of failure. By introducing greater redundancy in the star, it is possible to reduce the probability of such failures. In the simplest approach to providing such redundancy, each BR_a attached to \mathcal{X} peers with two RSs RS_1 and RS_2 .

In this arrangement, each BR_a receives one set of routes from each RS; we see from Eq. (11) that both the sets ${}^{o1}R_a$ and ${}^{o2}R_a$, respectively the views for BR_a maintained by RS_1 and RS_2 , are identical. We can easily show that this arrangement is functionally equivalent to the complete mesh. Intuitively, each BR_a receives two copies of the routes it would have received in the complete star; this does not change the contents of sS_a . When either RS fails, the ar-

angement reverts to the complete star. This solution does not require any coordination between RSs.

7.2. Load splitting

Multiple RSs may also be used to reduce RS storage requirements. Consider the arrangement at exchange point \mathcal{X} in which approximately half the BRs peer with one RS (denote this by RS_1) and the other half peer with a second RS, RS_2 . Each RS maintains a view for *each* of the the BRs on \mathcal{X} , not just those that directly peer with it. Each of these views is *partial*, in that the view for BR_x in RS_1 only contains the contributions of BRs peering with RS_1 .

Suppose now that BR_a peers with RS_1 . RS_1 computes partial view ${}^{o1}R_a$ using Eq. (11) over all BRs directly connected to it. In parallel, RS_2 computes partial view ${}^{o2}R_a$ using Eq. (11) for each of its peers. Then, RS_2 transfers the set of routes (by some unspecified mechanism) ${}^{o2}R_a$ to RS_1 , which merges these two partial views to get oR_a .

This ‘‘load splitting’’ approximately halves the storage requirements at an RS. In addition, it also halves RS processing requirements. This approach easily generalizes to more than two RSs; in the limit, when the load is split as many ways as there are BRs, the arrangement reverts to the complete mesh. Some inter-RS synchronization is necessary for exchanging partial views.

7.3. Destination splitting

A different approach to reducing RS storage requirements *stripes* the space of address prefixes across RSs. For example, each BR_a at an exchange point \mathcal{X} could peer with two RSs, RS_1 and RS_2 , but send routes to one half of the address space to RS_1 and send routes to the other half of the address space to RS_2 . Both RS_1 and RS_2 compute striped views ${}^{o1}R_a$ and ${}^{o2}R_a$ respectively, and advertise these to BR_a . BR_a then computes its set of selected routes from the union of the two striped views.

This solution also approximately halves the computational and storage complexity at each RS. The approach easily generalizes to more than two RSs. No inter-RS synchronization is necessary, but some

⁹ Even these ‘‘realistic’’ estimates are pessimistic. They are derived under the assumption that each BR *independently* chooses from among $j \times \rho(p)$ row functions. In practice, there is usually a correlation between the row functions for a destination. For example, two BRs may contract with the provider domain for d to select similar row functions for d . In this case, view decomposition can provide even greater benefits.

additional configuration may be necessary at each BR.

8. Related work

Inter-domain Policy Routing (IDPR) [17] is an architecture for source-specified policy routing. When using IDPR to send data to different address prefixes, each domain uses a domain-level source route satisfying its policy requirements. An IDPR “route server” computes this source route on behalf of a “client” domain, using global link-state topology information. Unlike an IDPR route server, our RS computes hop-by-hop routing tables satisfying its client domains’ policy requirements.

Like IDPR, the Unified inter-domain routing architecture [5] also uses the term “route server” to denote an entity that computes domain-level source routes (or fragments thereof) on behalf of domain BRs. Unified has two components: (1) A hop-by-hop routing component for realizing commonly expressed domain policies using IDPR, and (2) a source routing component for realizing more specialized policy using the Explicit Routing Protocol (ERP [6]) – these specialized policies are characterized by not being shared across enough domains to justify computation by the hop-by-hop component. Our RS, while clearly a part of the hop-by-hop component, can function as a Unified route server; to craft a domain-level source route to an address prefix, a domain BR can query the RS for the path vectors associated with IDPR routes to that address prefix [18].

The term “route server” is also used, in a slightly different sense, in the OSPF intra-domain routing

protocol specification [11]. In OSPF, domain BRs inject external routing information into intra-domain routing. Domain policies, usually configured into each domain BR, govern this dissemination of external routing information into OSPF. An alternative mechanism for injecting external routes into OSPF uses a single “route server” which, configured with domain policy, performs the same function. This use of the route server simplifies domain routing configuration.

The RIPE Route Server [1] is a precursor to our RS. RIPE’s Route Server is also designed to compute inter-domain routing tables, on behalf of domain BRs attached to an exchange point. However, their design assumes that there exists a single “preferred” path to an address prefix from the exchange point at which the RS is deployed. With this assumption, it suffices for their RS to compute one view for all attached BRs at the exchange point. Our more general design allows different domain BRs to use different paths from the exchange point to the same destination.

9. Conclusions

In this paper, we developed a formalism for describing the behavior of IDPR speaking BRs, stated a condition for functional equivalence between the mesh and the star, and used that to derive RS behavior in star peering. We also showed that techniques such as view decomposition can, under certain circumstances, potentially reduce RS storage requirements by a factor of five in some cases. By slightly relaxing the condition for functional equivalence, it is possible to extend view decomposition

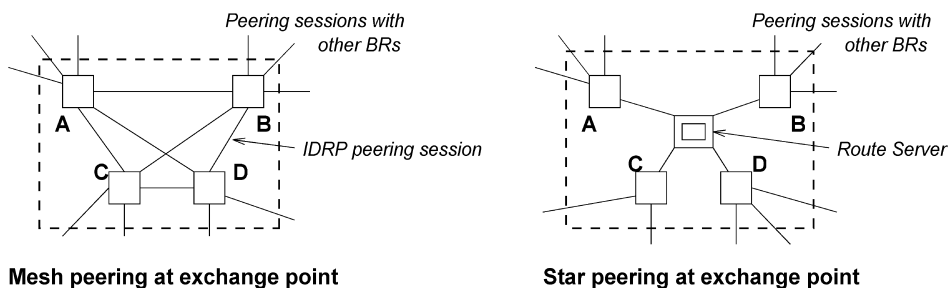


Fig. 7. RS behavior derivation: This figure is used to explain the proof for the converse of the RS behavior derivation of Section 3.

techniques to further reduce RS storage requirements; the paper does not discuss such approaches.

As described in Section 8, our RS can also be used to compute specialized domain-level source routes or source-route fragments in the Unified [5] architecture. Apart from using the path vectors contained in IDRP routes for this purpose, the RS can also compute these specialized source routes directly from information maintained in the Routing Arbiter Routing Registry.

Finally, because IDRP uses transport-level connections for route exchange, an RS need not be physically located at the exchange point it serves. This feature may be used to improve the failure tolerance of the complete star. More generally, this enables a number of variants of the hop-by-hop routing infrastructure; for example, a single RS could remotely serve more than one exchange point, a hierarchy of Route Servers could serve all inter-domain BRs in the Internet, and so on. Such an arrangement requires stable connectivity between a remote RS and the exchange point(s) it serves; frequent connectivity failures can impair the ability of inter-domain routing to quickly adapt to topology changes.

Appendix A. RS behavior derivation

This section sketches a proof for the following claim: If RS behavior at an exchange point \mathcal{L} is defined by Eq. (11) and the condition on $P_{a,RS}$ in Section 4.1, and the external inputs (i.e., those from peers not on \mathcal{L}) to the BRs at \mathcal{L} are the same in both the mesh and the star, then the equivalence condition of Section 3 (that is, all BRs at \mathcal{L} select the same routes in both arrangements) holds. This claim is the converse of the derivation described in Section 4.1.

Suppose that to some destination d , BR A in Fig. 7 selects route $r_{\#}$ in the mesh and route r_{\star} in the star. Suppose, to the contrary, that r_{\star} is different from $r_{\#}$. Since all external inputs to A are the same in both arrangements, A must have selected r_{\star} from oR_a .

This implies that at least one of the BRs, say B , must be advertising a different route to d in the star than in the mesh (Eq. (11)). Therefore, B must have

selected different routes to d in the two arrangements. Repeating the argument in the above paragraph, at least one of A , C , or D must have selected different routes to d in the two arrangements. Clearly, it cannot be A otherwise we arrive at a circularity; thus, B selects different routes because one of C or D selects different routes in the two arrangements. Proceeding thus, we arrive at a contradiction.

References

- [1] T. Bates, Technical implementation of the RIPE route server, in: Proc. INET'93, San Francisco, CA, August 1993.
- [2] T. Bates, E. Gerich, L. Joncheray, J.-M. Jouanigot, D. Karrenberg, M. Terpstra, J. Yu, Representation of IP routing policies in a routing registry, Technical Report ripe-181, RIPE NCC, Amsterdam, Netherlands, October 1994.
- [3] Protocol for Exchange of Inter-domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs, ISO/IEC JTC1/SC6 CD10747.
- [4] D. Estrin, J. Postel, Y. Rekhter, Routing arbiter architecture, *ConneXions* 8 (8) (August 1994) 2–7.
- [5] D. Estrin, Y. Rekhter, S. Hotz, A scalable inter-domain routing architecture, in: Proc. ACM SIGCOMM'92, Baltimore, MD, August 1992, pp. 40–52.
- [6] P. Ford, T. Li, Y. Rekhter, Explicit Routing Protocol for IPv6, Internet-Draft, work in progress. Available from <ftp://ds.internic.net/internet-drafts/draft-ietf-sdrp-erp-00.txt>.
- [7] P. Gross, P. Almquist, IESG Deliberations on Routing and Addressing, Request for Comments 1380, Internic Directory Services, <ftp://ds.internic.net/rfc>, November 1992.
- [8] S. Heimlich, Advanced networks and services, Private communication.
- [9] General information about RIPE and the RIPE NCC, Technical Report ripe-057, RIPE NCC, Amsterdam, Netherlands, October 1994.
- [10] MERIT Network Inc., NSFNET Network Announcement Change Request v7.1, MERIT Network Inc., Ann Arbor, Michigan. Available from nic.merit.edu/~nsfnet/announced.networks/template.net.README, February 1994.
- [11] J. Moy, OSPF Version 2, Request for Comments 1583, Internic Directory Services, <ftp://ds.internic.net/rfc>, March 1994.
- [12] Network Access Point Manager, Routing Arbiter, Regional Network Providers and Very High-Speed Services Provider for NSFNET and the NREN (SM) Program, National Science Foundation Program Solicitation 93-52, May 1993.
- [13] NSFNET Policy Routing Database (PRDB), Maintained by MERIT Network Inc., Ann Arbor, Michigan. Contents available from <ftp://nic.merit.edu/nsfnet/announced.networks/nets.tag.now>.

- [14] Y. Rekhter, Inter-Domain Routing Protocol (IDRP), *IEEE/ACM J. Internetworking* 3 (1993).
- [15] Y. Rekhter, T.Li, A Border Gateway Protocol 4 (BGP-4), Request for Comments 1654, Internic Directory Services, <ftp://ds.internic.net/rfc>, July 1994.
- [16] RIPE Policy Database, Contents available from <ftp://ftp.ripe.net/ripe/dbase/ripe.db>.
- [17] M. Steenstrup, An Architecture for Inter-Domain Policy Routing, Request for Comments 1478, Network Information Center, July 1993, T.J. Watson Research Center, IBM Corp.
- [18] K. Varadhan, D. Estrin, S. Hotz, Y. Rekhter, SDRP Route Construction, Internet-Draft, work in progress, Available from <ftp://ds.internic.net/internet-drafts/draft-ietf-sdrp-route-construction-00.txt>.