An Algebraic Approach to Computing the Reliability of Internet Routing

Qi Li*, Mingwei Xu*, Jianping Wu*, Patrick P. C. Lee‡, Ke Xu*
*Dept. of Computer Science, Tsinghua University, †Tsinghua National Lab for Information Science and Technology
‡Dept. of Computer Science and Engineering, The Chinese University of Hong Kong
{liqi, xmw, jianping, xuke}@csnet1.cs.tsinghua.edu.cn, pclee@cse.cuhk.edu.hk

***Abstract*—Evaluating the reliability of Internet routing is important for an ISP to assess existing peer relationships or establish new peer relationships. Existing algorithms for network reliability computations take all routing paths as inputs. However, these paths may not be actually available for routing because of the constraints of routing policies in the Internet. In this paper, we propose an algebraic approach that effectively reduces the number of candidate routing paths according to the given routing policy. We further improve the accuracy of the routing reliability result by subtracting the miscounted value of routing paths due to overlooking routing policy constraints.**

I. Introduction

The computation of routing reliability plays an important role in Internet routing, where a high degree of reliability is crucial. Extensive efforts have been placed in the development of mathematical models and efficient computation techniques for network reliability of general graph structures [1], [13]. However, few studies were done to specifically address the reliability of Internet routing. The Internet is connected by *policy-based routing protocols*, e.g., Border Gateway Protocol (BGP) [14], between different autonomous systems (ASes), but such routing protocols are generally prone to policy conflicts [5], [4] which incur serious reliability problems. We argue that understanding the routing reliability is important for an ISP to assess existing peer relationships and establish new peer relationships. Thus, we advocate that it is essential to develop a fundamental understanding of routing reliability in the Internet, especially in the presence of routing policy conflicts.

In traditional network reliability computation algorithms, all edges[1] are used for reliability computations. However, it is not straightforward to directly apply such algorithms for Internet routing, which is policy-based. Not all edges imply network connectivity, as the underlying routing policies may not include some of the edges in routing computations. To illustrate, Figure 1 shows the complications of the problem. For example, routing path $\{1\}$ cannot be used to compute the reliability between vertices a and c because vertex c sets the preference of vertex a (i.e., the routing path $\{1\}$) to 0[2] and will not adopt it as a candidate routing path. In addition, we need to aggregate reliability computations if upstream nodes have multiple routing paths to destinations. For instance, in Figure 1, vertex b has two routing paths to vertex a. Vertex c needs to use the aggregated reliability values between vertex b and vertex a to compute its own reliability since vertex c will not know all routing paths in vertex b at the same time because of the routing policies. Moreover, vertex c and vertex d simultaneously set a higher preference to each other, and they cannot form valid routing paths when they adopt routes that traverse each other and then the reliabilities from these vertices to vertex a at this stage should be equal to 0. Such complications make reliability computations become a challenging problem. There are very few studies which address policy-based routing and study the theoretical reliability of Internet routing.

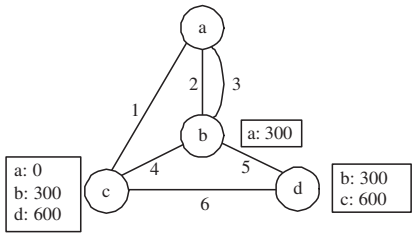

Fig. 1. An example topology that uses policy-based routing. Each value refers to the routing preference. A higher value means higher preference, and a zero value means the path is not chosen.

In this paper, we propose an algebraic approach to identifying available routing paths and then computing the network reliability with respect to the routing policies. We call the network reliability considering routing policies to be the *routing reliability*. We adopt the concept of routing algebra to identify all available edges for routing reliability computations. Our algebraic approach is to show how multiple routes can be propagated and how the algebraic approach can be extended to accurately compute routing reliability. The approach effectively reduces the computation complexity by filtering out the routing paths that are not included in the routing policies, and aggregating the routing paths that are not simultaneously available. We extend the routing algebra by introducing a new parameter, CAUSE, in route signatures in routing algebra to identify the set of conflicting routing paths. Thus, we accurately compute routing reliability by subtracting

[1]In this paper, "edge" and "link", "path" and "edge set" are used interchangeably.

[2]In reality, a routing path is disabled by filtering in policy-based routing. For simplicity, we assume that the paths are disabled by setting the *local preference* to 0.

978-1-4577-0638-7 /11/$26.00 ©2011 IEEE

the miscounted value of conflicting routing path set, so as to improve the accuracy of the result.

The rest of the paper is organized as follows. In Section II, we briefly review the basic network reliability computation model. We propose an algebraic approach to identify available routing paths and accurately compute routing reliability in Section III. Section IV and V present the related work and conclude the paper, respectively.

II. Network Reliability Computation

Network reliability can be computed using a system reliability model of a *coherent system* [1], [12]. In this system, let x_i be a binary variable that indicates the state of i-th component (e.g., node, link) in the system, and let ϕ be the binary variable that indicates the state of the network based on the states of components. If $x_i = 1$ ($\phi = 1$), then it means the i-th component (the system) is functioning; otherwise, $x_i = 0$ ($\phi = 0$). We can obtain ϕ by calculating $\phi(\mathbf{x})$, where $\mathbf{x}$=($x_1 \ldots x_n$) is the vector of the states of all components, n is the number of components in the network, and function $\phi(\mathbf{x})$ is called the *structure function*. To compute network reliability, we should generate the minimal path vectors or the minimal cut vectors of the network, defined as follows.

Definition 1: (**Path Vector**) A path vector is a vector $\mathbf{x}$ such that $\phi(\mathbf{x})$ =1.

Definition 2: (**Cut Vector**) A cut vector is a vector $\mathbf{x}$ such that $\phi(\mathbf{x})$ =0.

Given two vectors $\mathbf{y}$ and $\mathbf{x}$, $\mathbf{y} < \mathbf{x}$ means $\forall i$, $y_i \leq x_i$ (i=1,. . .,n), and $\exists j$, $y_j < x_j$ ($1 \leq j \leq n$).

Definition 3: (**Minimal Path Vector**) A path vector $\mathbf{x}$ is a minimal path vector if $\phi(\mathbf{y}) = 0$, $\forall$ $\mathbf{y} < \mathbf{x}$ where $\phi(\mathbf{x}) = 1$.

The path set C_1=$\{i|x_i$=1$\}$, which we call a *minimal path set*, constitutes a minimal path set of elements whose functioning ensures network connectivity. We denote the j-th minimal path set as P_j, where j=1,. . .,m and m is the number of minimal path sets of ϕ.

Definition 4: (**Minimal Cut Vector**) A path vector $\mathbf{x}$ is a minimal cut vector if $\phi(\mathbf{y}) = 1$, $\forall$ $\mathbf{y} > \mathbf{x}$ where $\phi(\mathbf{x}) = 0$.

The path set C_0=$\{i|x_i$=0$\}$, which we call a *minimal cut set*, constitutes a minimal cut set of elements whose failures will disconnect the network. We denote the j-th minimal cut set as K_j, where j=1,. . .,m and m is the number of minimal cut sets of ϕ.

Suppose that for state X_i of the i-th components,

$$P[X_i = 1] = p_i = EX_i \quad for\ i = 1 \ldots n, \tag{1}$$

where EX denotes the expected value of the random variable X, and p_i is the probability that component i functions, i.e., the reliability of i. The reliability of the system is thus given by:

$$P[\phi(X) = 1] = h = E\phi(X). \tag{2}$$

The system reliability h is a function of component reliability, such that

$$h = h(\mathbf{p}), \tag{3}$$

where $\mathbf{p}$=(p_1, ..., p_n) present the vector of component reliabilities and $h(\mathbf{p})$ is referred to as the *reliability function of structure* ϕ. Thus, the reliability functions of a series structure $\phi(\mathbf{x}) = \prod_{i=1}^{n} x_i$ or a parallel $\phi(\mathbf{x}) = \coprod_{i=1}^{n} x_i$ can be respectively computed as follows:

$$h(\mathbf{p}) = \prod_{i=1}^{n} p_i, \text{ and} \tag{4}$$

$$h(\mathbf{p}) = \coprod_{i=1}^{n} p_i = 1 - \prod_{i=1}^{n} (1 - p_i). \tag{5}$$

The network reliability can be computed by expanding the above system function state[3] represented by either minimal path or cut sets into the x_i's multinomial expressions, using the idempotency of x_i (i.e., x_i^2=x_i), and taking the expectation as follows:

$$h(\mathbf{p}) = E \coprod_{j=1}^{m} \prod_{i \in P_j} X_i. \tag{6}$$

$$h(\mathbf{p}) = E \prod_{j=1}^{m} \coprod_{i \in K_j} X_i. \tag{7}$$

In a network, a vertex may have different routing paths with different preferences to a destination [14]. Sum of Disjoint Products (SDP) [13] can be extended to address the issue of reliability calculations under multiple paths with different preferences. In this paper, for simplicity but without loss of generality, we only introduce the basic reliability model to compute the routing reliability.

III. Identifying Routes for Routing Reliability Computations

By modeling a network as a directed graph, we can calculate the network reliability based on the graph structure. From the point of view of Internet routing reliability, however, we can not directly apply the network reliability theory since policy routing in the Internet does not allow all edges in the graph to provide network connectivity. In this paper, we adapt the basic routing algebra defined by Sobrinho [16] to identifying all available edges for routing reliability computation. We point out that although Sobrinho defines a basic routing algebra, a complete policy routing algebra with *local preference* is not defined. Here, we describe the Internet topology with *local preference* as an algebra[4]. Generally, a policy routing algebra consists of an ordered 6-tuple

$$(L, \Sigma, W, f, \preceq, \oplus)$$

which comprises:

- a set of $labels$ L;
- a set of $signatures$ Σ;

[3]In network(/routing) reliability, system function state in this paper means the function state of connectivity between different nodes.

[4]Flavel et al. extended Sobrinho's to an iBGP algebra [4]. We do not discuss the iBGP issue in this paper since we focus on routing reliability computations between different ISPs/ASes.

- a set of $weights$ W;
- a function f that maps signatures into weights;
- a total order $\preceq$ on W; and
- a binary operation $\oplus$ that maps pairs of a label and a signature into a signature, i.e., $\oplus$: $L \times \Sigma \rightarrow \Sigma$.

A basic routing algebra comprises a set of labels, a set of signatures, and a set of weights. Each network link has a label and each network path has a signature. In the algebra, the set of labels L contains all feasible edge labels for a topology. Labels are simply all edges, e.g., they are the set of links in the topology, $L = \{l\}$. The set of signatures Σ describes all feasible routes. On the other hand, in policy routing, if Σ only represents a set of edges, i.e., $\Sigma \subseteq 2^L$, a route has a signature solely based on the number of edge labels to reach the destination. That is, $\Sigma = \mathbb{L}^+$, where $\mathbb{L}^+$ indicates a non-empty ordered edge set. The lower the weight of a signature the more preferred signature is.

Sobrinbo showed that convergence of a policy routing protocol is guaranteed if the algebra is strictly monotone [16]. An algebra for routing is strictly monotone if for all $l \in L$ and $\sigma \in \Sigma - \emptyset$, $f(\sigma) \prec f(l \oplus \sigma)$ where $\prec$ indicates a strict preference of the former signature over the later one, which means preference of a routing path strictly decreases when it is propagated.

A. Basic Reliability Computation

In policy routing, each edge has a *Local Preference* (PREF) to indicate ASes' preference for its multiple edges. Thus, L is defined as the lexical cross product of all edge sets and their PREF values. For simplicity, we assume that edge reliability is indicated in the edge set. That is,

$$L = \underset{\text{PREF}}{\mathbb{Z}^+} \times \underset{\text{edge}}{\{1\}}$$

We can extend the set of signatures which defined as the cross product of all edge sets of routing path and their PREF attributes.

$$\Sigma = \underset{\text{PREF}}{\mathbb{Z}^+} \times \underset{\text{edge set}}{\mathbb{L}^+}$$

In policy routing, vertices propagate their chosen routes to neighbors. In the policy routing algebra, this process is undertaken by the $\oplus$ operator. The $\oplus$ operator takes a signature and an edge label as inputs, and returns a signature. In general, in policy routing, an AS(/router)'s selected signature is an ordered set of edge labels with PREF which is not transitive in the algebra. This signature together with an edge label is combined to create a new signature that represents a new set of edge labels with PREF. The sum of the existing signature and the edge label can be expressed as σ = (PREF, $\mathbb{L}^+$) $\subseteq$ Σ, and then the new signature is σ'=(PREF', l)$\oplus\sigma$=(PREF', l)$\oplus$(PREF, $\mathbb{L}^+$)=(PREF', $\mathbb{L}^+ \cup l$).

Figure 2 shows a partial topology extracted from a backbone network with generated routing policies enforced. The link reliability shown in Figure 2 is normalized from the failure number of different links in November 2008 [8]. Now let us consider calculating the signatures between vertex c and vertex a in Figure 2. Vertex c has signatures to vertex a: σ_c^b = (300, {2}) $\oplus$ σ_b = (300, {2}) $\oplus$ (300, {1}) = (300, {2, 1}), and σ_c^d = (300, {2}) $\oplus$ σ^d = (300, {4}) $\oplus$ (1200, {3}) $\oplus$ (300, {1}) = (300, {4}) $\oplus$ (1200, {3, 1}) = (300, {4,3,1}). Note that if vertices have multiple signatures to destinations and these signatures may not visible to downstream vertices simultaneously, we should aggregate reliability computations of these signatures in these vertices. For example, vertex c has two multiple signatures and then vertex e should aggregate the signatures in vertex c as (300, {5,$c\rightarrow a$}). That is, to compute routing reliability of $e\rightarrow a$, vertex e does not need to use two signatures available in vertex c, i.e., (300,{4,3,1}) and (300,{2,1}), but directly uses the reliability of $c\rightarrow a$. The computed signatures in each vertex are shown in Table I where σ^1 and σ^2 indicate the computed signatures according to signatures from two different neighboring vertices.

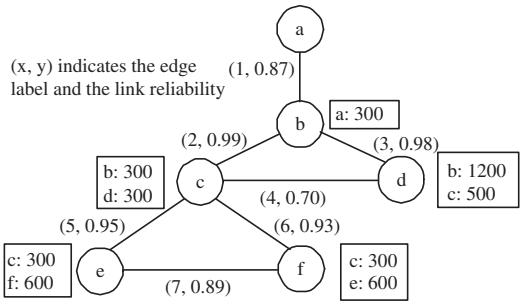

Fig. 2. A partial topology extracted from a backbone network with generated routing policies. Note that the routing policies enforced are used to demonstrate the effectiveness of routing reliability computations.

Σ	σ^1	σ^2
$b\rightarrow a$	(300,{1})	$\emptyset$
$c\rightarrow a$	(300,{4,3,1})	(300,{2,1})
$d\rightarrow a$	(1200,{3,1})	(500,{4,2,1})
$e\rightarrow a$	(600,{7,6,$c\rightarrow a$})	(300,{5,$c\rightarrow a$})
$f\rightarrow a$	(600,{7,5,$c\rightarrow a$})	(300,{6,$c\rightarrow a$})

TABLE I
THE COMPUTED Σ FOR EVERY VERTEX IN FIGURE 2

A vertex in routing always has multiple signatures it can select, and the selection process is to identify what it determines is the best signature. Route signatures always contain multiple attributes, and route selection is based on the predefined criteria. We use the function f to convert a set of signatures to a set of weights W that are comparable using the operator $\preceq$. The preference of weights is *transitive*. For instance, if a is preferred to b and b is preferred to c, a must be preferred to c. Then, the function f simply returns the weight of the signature. The function f compares PREF and the number of labels such that signatures are preferred with the maximal PREF value prior to the number of labels to the destination,

Routing reliability computation in vertex x

Step 1: Obtain signature σ_j for its neighboring vertex j connecting with edge l_j and compute σ_x^j=$l_j \oplus \sigma_j$;

Step 2: Extract the set of the edge set $\{L\}$ from each σ_x, $\sigma_x \in \Sigma_x$, if $f(\sigma_x) \neq (\infty, \infty)$ and assign each edge set in $\{L\}$ to the path sets P (P_1,P_2, ..., P_m);

Step 3: Compute the routing reliability with Equation (6) according to each edge reliability.

Fig. 3. Basic routing reliability computation algorithm

i.e.,

$$f(\sigma) = \begin{cases} (1/\text{PREF}, n), & \text{if } \sigma = (\text{PREF}, \mathbb{L}^+), \\ (\infty, \infty), & \text{if } \sigma = \emptyset, \end{cases} \quad (8)$$

where n = $number(\mathbb{L}^+)$ which indicates the number of edges in the non-empty edge set. We compare weights of different signatures lexicographically. That is, we first prefer a routing path with the highest PREF value, and if equal, prefer the routing path with the shortest routing path.

The routing reliability can be computed by expanding the system function state of edge sets, $h(\mathbf{p})$, in Σ. Let us follow the example shown in Figure 2. Note that routing policies in the topology are conflicting and some edges permitted by policies cannot be used to calculate routing reliability. Here, we only illustrate the basic procedure of routing reliability computation. We will present an accurate computation procedure to address the policy conflict issue in Section III-B. Figure 3 shows the process to compute routing reliability with *minimal path sets*.

Let us follow example shown in Figure 2 to illustrate how to compute route reliability. For simplicity, we only compute the routing reliability of the connectivity to vertex a. For example, for the paths from vertex c to vertex a, according to routing policies specified in Figure 2, we can directly obtain two edges, L=$\{l_b, l_d\}$ where l_b=(300, 2) and l_d=(300, 4). As we assume that routes are propagated in the network, all available routing paths can be calculated by $\oplus$, we can obtain the signatures Σ_c=$\{\sigma_c^b$,$\sigma_c^d\}$ where σ_c^b=$l_b \oplus \sigma^b$=(300, 2)$\oplus$(300, {1})=(300, {2,1}) and σ_c^d = $l_d \oplus \sigma^d$=(300,4)$\oplus$(1200,{3,1})=(300, {4,3,1}) (σ_x^y indicates the signature in vertex x is learned from node y). The signatures of complete paths to vertex a are shown in Table I. According the definition of function f, we can obtain $f(\sigma^1) < f(\sigma^2)$ which means that routing paths specified in σ^1 are preferred over that in σ^2 (f indicates the preference of every signature and will be used in Section III-B to address the policy conflict issue).

According to the signatures illustrated in Table I, we can obtain the *minimal path sets* and then compute the routing reliability according to Equation (6). For instance, we can calculate minimal path sets from vertex c to vertex a according to Σ_c. Paths are specified by L in signatures, i.e.,P_1={4,3,1} and P_2={2,1}, and then we can obtain $h(\mathbf{p}) = E\coprod_{j=1}^{m}\prod_{i\in P_j} X_i$=0.9441 according to Equation (6). Moreover, as shown in Table I, the reliabilities of $e \rightarrow a$ and $f \rightarrow a$ rely on that of $c \rightarrow a$, and we use the reliability of $c \rightarrow a$ to compute them. Thus, we aggregate reliability computations during computing reliabilities of $e \rightarrow a$ and $f \rightarrow a$. Table II shows the routing reliability of the connectivity to vertex a in the network. For simplicity, in this paper, we compute the reliability from all vertices to vertex a. The overall routing reliability of the connectivity to vertex a is 0.9239, which is better than the average link reliability, 0.9014. Actually, if we compute the routing reliability by expanding the system function state represented by the *minimal cut sets*, we will achieve the same results as shown in Table II by arranging all edge labels in Σ to the *minimal cut sets*.

Paths	basic	link	difference
$b \rightarrow a$	0.8700	0.9014	-3.48%
$c \rightarrow a$	0.9441	0.9014	+4.73%
$d \rightarrow a$	0.8526	0.9014	-5.41%
$e \rightarrow a$	0.9775	0.9014	+8.44%
$f \rightarrow a$	0.9754	0.9014	+8.20%

TABLE II
THE RELIABILITY OF THE CONNECTIVITY TO VERTEX a.

The basic reliability results shown in Table II are the upper bounds of routing reliabilities. We may not obtain these values and the values may not be accurate in presence of policy conflicts since the basic routing reliability algorithm shown in Figure 3 does not consider policy conflicts in routing and some values are miscounted. Actually, we cannot fully utilize all edge sets seen by each vertex to compute routing reliability if there exist routing policy conflicts. In Section III-B, we will address this issue and obtain the accurate routing reliability by identifying policy conflicts in the routing algebra.

B. Accurate Routing Reliability

We have provided an algebraic approach to compute the routing reliability. However, the routing algebra is not monotone and the policy routing protocol does not converge [16]. The routing reliability is still not perfect since some routing paths still cannot be used while the routing policies allow these paths. For example, as shown in Figure 2, initially vertex e and vertex f just choose signatures (300,{5,$c \rightarrow a$}) and (300,{6,$c \rightarrow a$}). After it, vertex e changes its chosen signature to (600,{7,5,$c \rightarrow a$}) because $f(l \oplus \phi)$= $f(600, \{7, 5, c \rightarrow a\}) \prec f(\phi)$=$f(300, \{5, c \rightarrow a\})$. Vertex f has a similar behavior. Vertex e will learn (600,{7,5,$c \rightarrow a$}) from vertex f and its signature will revert to (300,{5,$c \rightarrow a$}) since choosing the received signature {7,5,$c \rightarrow a$} will cause a path loop. Similarly, the chosen signature of vertex f will revert to initial state. The signatures from vertex e and vertex f to vertex a are keeping changing and cause routing loops. Thus, the connectivity from these vertices (and downstream vertices) to vertex a have sever reliability problem. As shown in Figure 2, the reliabilities of the connectivity from vertex e and vertex f to vertex a should be 0, i.e., $\phi(\mathbf{x})$=0, because they do not have valid paths to the destination a. Therefore, we should consider these cases during reliability computations.

To address this issue, we introduce to the route signature a new parameter called CAUSE that denotes the cause of routing changes to the vertex that triggers routing changes in Internet

routing. The parameter CAUSE will specify a link l'(x,y) in a signature in vertex y indicating the signature in vertex y is from vertex x via edge l'. The parameter will be set in vertices only when $f(\sigma)<f(\sigma')$ and $n(\sigma)>n(\sigma')$ ($\sigma,\sigma'\in\Sigma$ and $n(\sigma)$ means the edge number in σ) where $f(\sigma')<f(\sigma^*)$ for $\forall\sigma^*\in\Sigma$-σ-σ', which indicates that the potential existence of routing policy conflicts [5]. In most cases, CAUSE equals to $\emptyset$. For example, in Figure 2, CAUSE in the signatures of vertices a, b, and c is $\emptyset$, and vertices e and f will generate CAUSE 7(f,e) and 7(e,f), respectively. In vertices e and f, $f(\sigma_x)$=1/600$<f(\sigma'_x)$ and $n(\sigma_x)$=4$>n(\sigma'_x)$ where $f(\sigma'_x)<f(\sigma^*_x)=(\infty,\infty)$, $\forall\sigma^*_x\in\Sigma_x$-$\sigma_x$-$\sigma'_x$ (x=e,f). Here, we call conflicting signatures existing in vertices e and f. The generated CAUSE values will be attached to the signatures along the traversed edges. The edge labels remain unchanged, but the set of signatures becomes as follows.

$$\begin{array}{ccccccc} \Sigma & = & \mathbb{L}^+ & \times & \mathbb{Z}^+ & \times & \mathbb{L}^+ \\ & & \uparrow & & \uparrow & & \uparrow \\ & & \text{CAUSE} & & \text{PREF} & & \text{edge set} \end{array}$$

For accurate routing reliability computation, we should substrate the reliability values of the set of conflicting routing paths from the entire reliabilities. It is easy for vertices to detect conflicting paths by checking if their signatures include edges specified by CAUSE. If the lowest weight signature, σ, (i.e., $f(\sigma)<f(\sigma^*)$ if $\sigma\in\Sigma$ and $\forall\sigma^*\in\Sigma$-σ), contains non-empty CAUSE, it means the weight of signature is not monotonically increase with the increase in the number of edges in signature and the chosen signature will be changed between these two lowest weight signatures, σ and σ', (i.e., $f(\sigma)<f(\sigma')$ where $f(\sigma')<f(\sigma^*)$ for $\forall\sigma^*\in\Sigma$-σ-σ'). The reliability value should be equal to 0 at this stage but is miscounted in the basic reliability computation (see Section III-A). Thus, we should substrate this miscounted value from the overall reliability. The miscounted value is equal to the product of the reliability of each edge in the union of the edge sets specified in two lowest weight signatures, i.e., σ and σ', in each vertex where CAUSE$\in\sigma$. Note that CAUSE$\in\sigma$ means that the edge specified by CAUSE is included in the edge set of σ. We can revise Equation (6) to accurately calculate the reliability accord to *minimal path*.

$$h(\mathbf{p}) = \begin{cases} E\coprod_{j=1}^{m}\prod_{i\in P_j}X_i - E\prod_{k\in F}Y_k, & \text{if } F\neq\emptyset, \\ E\coprod_{j=1}^{m}\prod_{i\in P_j}X_i, & \text{otherwise,} \end{cases} \tag{9}$$

where F=$\bigotimes_{i=1}^{m}P_i$ indicates a set of edges, $P_i(1\leq i\leq m)$, which unions the edge set specified in two lowest weight signatures, if CAUSE in the lowest weight signature is not empty and the edge set in the signature contains the edge specified by CAUSE; Y_k indicates k-th component in F. Note that Equation (9) only considers the case that there only exists single policy conflict between two lowest weight signatures, which has received much attention [5], [7], [3], [10], [15], [2]. We can easily extend the equation to compute routing reliability under complicated policy conflict cases, e.g., a network failure breaks one policy conflict but raises another conflict. Figure 4 shows the process to accurately compute routing reliability with *minimal path sets*.

Accurate routing reliability computation in vertex x

Step 1: Obtain signature σ_j for its neighboring vertex j connecting with edge l_j and compute σ_x=$l_j\oplus\sigma_j$;

Step 2: Extract the set of the edge set $\{L\}$ from each σ_x, $\sigma_x\in\Sigma_x$, if $f(\sigma)\neq(\infty,\infty)$ and assign each edge set in $\{L\}$ to the path sets P (P_1,P_2, ..., P_m);

Step 3: Compute $E\coprod_{j=1}^{m}\prod_{i\in P_j}X_i$ according to each edge reliability;

Step 4: Extract $\{L, L'\}$ from σ_x and σ'_x and assign to F if CAUSE$\in\sigma_x$ and $f(\sigma_x)<f(\sigma'_x)<(\infty,\infty)$ where $f(\sigma'_x)<f(\sigma^*_x)$ for $\forall\sigma^*_x\in\Sigma_x$-$\sigma_x$-$\sigma'_x$;

Step 5: Compute $E\prod_{k\in F}Y_k$ and obtain the final routing reliability with Equation (9).

Fig. 4. Accurate routing reliability computation algorithm

Let us follow the example shown in Figure 2. Vertex e and vertex f prefer the signatures from each other, i.e., (600,{7,6,$c\rightarrow a$}) and (600,{7,5,$c\rightarrow a$}). However, (600,{7,6,$c\rightarrow a$}) and (600,{7,5,$c\rightarrow a$}) are conflicting, and the generated CAUSE is 7(f,e) and 7(e,f) in vertex e and vertex f, respectively. Here, we still compute the routing reliability using the *minimal path sets*. For vertex e, the signature (600,{7,6,$c\rightarrow a$}) contains the edge 7(f,e) specified by CAUSE, and F is equal to the union of edge sets in two lowest weight signatures, (600, {7,6,$c\rightarrow a$}) and (300,{5,$c\rightarrow a$}), i.e., F={7,6,$c\rightarrow a$,5,$c\rightarrow a$}. Let us compute the product of the expectation of edges in F: $E\prod_{j\in F}Y_j$=0.7863. The minimal path sets from vertex e to vertex a is P_1={7,6,$c\rightarrow a$} and P_2={5,$c\rightarrow a$}. Then, we obtain the routing reliability from vertex e to vertex a: $h(\mathbf{p}) = E\coprod_{j=1}^{m}\prod_{i\in P_j}X_i$-$E\prod_{j\in F}Y_j$=0.9774-0.7863=0.1651. Similarly, we can obtain the routing reliability of $f\rightarrow a$, and the result is 0.1890. The reliabilities are much lower than that achieved in Section III. The overall routing reliability of the connectivity to vertex a is 0.6094. Compared to the basic routing reliability computed in Section III-A, the accurate routing reliability is reduced by 33.29%. The detailed results are shown in Table III.

Paths	accurate	basic	difference
$b\rightarrow a$	0.8700	0.8700	0%
$c\rightarrow a$	0.9441	0.9441	0%
$d\rightarrow a$	0.8526	0.8526	0%
$e\rightarrow a$	0.1911	0.9014	-80.44%
$f\rightarrow a$	0.1890	0.9014	-80.62%

TABLE III
THE ACCURATE RELIABILITY OF THE CONNECTIVITY TO VERTEX a.

Similarly, we can rewrite Equation (7) and calculate the routing reliability with the *minimal cut sets* to obtain the same results of routing reliability.

$$h(\mathbf{p}) = \begin{cases} E\prod_{j=1}^{m}\coprod_{i\in K_j}X_i - E\prod_{j\in F}Y_j, & \text{if } F\neq\emptyset, \\ E\prod_{j=1}^{m}\coprod_{i\in K_j}X_i, & \text{otherwise,} \end{cases} \tag{10}$$

where F=$\bigotimes_{i=1}^{m} P_i$ indicates a set of edges, $P_i (1 \leq i \leq m)$, which unions the edge set specified in two lowest weight signatures, if CAUSE in the lowest weight signature is not empty and the edge set in the signature contains the edge specified by CAUSE; Y_k indicates k-th component in F. Here, we do not repeat the reliability computations.

According to the accurate routing reliability results shown in Table III, we can observe that the basic and accurate routing reliability values will be equal if there does not exist any policy conflicts between themselves. For example, with two different route reliability computation algorithms, the reliabilities of the connectivity from vertices b, c, and d to vertex a are equal (see Table III). However, If a policy conflict exists, the accurate values are much worse than the basic values, e.g., in vertices e and f, though they have multiple parallel connections to vertex a. In next section, we will solve policy conflicts between different vertices by modifying the weight function in the algebra, and then we can fully utilize connections seen by each vertex to improve the routing reliability.

IV. Related Work

Network reliability can be calculated as a system reliability of a coherent system [1]. The theory is used to calculate the basic network reliability without considering routing policies enforced in networks. Sum of disjoint products (SDP) approaches are proposed to reduce computation costs and the storage space [13], [11]. SDP converts the sum of products into the sum of disjoint products which has a one-to-one correspondence with the reliability expression [11]. Different SDP algorithms are summarized and compared in [13]. Our routing reliability approach is orthogonal to these algorithms. We can leverage these SDP algorithms to reduce computation complexity.

Various algebras are proposed to prove properties of routing, such as stability, safety, and robustness. Sobrinho proposed a basic policy routing algebra to explore design principles towards the creation of safe policy routing [16]. Griffin and Sobrinho developed a unified algebraic framework to identify fundamental properties a vector or link-state routing protocol ensuring correct behaviors [6]. Flavel et al. extended Sobrinho's to an iBGP algebra [4] and proposed some new attribute to the basic routing algebra so as to ensure strict monotonicity of iBGP algebra. Recently, Le *et al.* proposed a new set of connecting primitives to the algebra to ensure safety of routing across multiple routing instances [9]. In the paper, we propose an eBGP algebra to model Internet routing and ensure strict monotonicity of the algebra for safe routing reliability computations.

Routing instability caused by conflicting routing policies is well studied in literature by analyzing abstract models of BGP [5], [7], [3], [10], [4], [15], [2]. Several proposals were presented to solve this issue. Griffin *et al.* suggested a BGP extension called *simple path vector protocol (SPVP)* [5], which detects cycles in route selections by carrying route selection history within each routing update. Different from SPVP, Ee *et al.* [2] detect different routing policy conflict cases through the use of history tables. Routes received from neighbors and the sequence numbers of the incoming routes are stored, so that routing policy conflicts can be detected by comparing the sequence numbers. Our algebraic framework points out a new approach to address conflicting routing policies by identifying loopy route selections.

V. Conclusion

In this paper, we propose an algebraic approach for routing reliability computations in Internet routing. We identify all available routing paths and leverage the traditional algorithms of network reliability to compute routing reliability. We improve the accuracy of routing reliability by extending the routing algebra to identify the set of conflicting routing paths and substrating the miscounted values of routing reliability. In future work, we will revise the routing algebra to ensure strict monotonicity of the algebra such that we can fully utilize all available signatures to improve the routing reliability.

Acknowledgements

This work is supported by NSFC grant No. 61073166, 973 Program grant No. 2009CB320502, 863 Program grant No. 2009AA01Z251, and the National Science & Technology Pillar Program of China, grant No. 2008BAH37B03.

References

[1] R. Barlow. *Statistical theory of reliability and life testing: probability models*. Holt, Rinehart and Winston, New York, 1974.

[2] C.T. Ee, V. Ramachandran, B. Chun, K. Lakshminarayanan, and S. Shenker. Resolving inter-domain policy disputes. In *Proceeding of ACM SIGCOMM*, pages 157–168, 2007.

[3] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. In *Proceeding of ACM SIGCOMM*, pages 25–36, 2005.

[4] A. Flavel and M. Roughan. Stable and flexible iBGP. In *Proceeding of ACM SIGCOMM*, pages 183–194, 2009.

[5] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[6] T. G. Griffin and J. Sobrinho. Metarouting. In *Proceeding of ACM SIGCOMM*, pages 1–12, 2005.

[7] T. G. Griffin and G. Wilfong. Analysis of the med oscillation problem in BGP. In *Proceeding of IEEE ICNP*, pages 90–99, 2002.

[8] M. Hou, D. Wang, M. Xu, and J. Yang. Selective protection: A cost-efficient backup scheme for link state routing. In *Proceedings of the IEEE ICDCS*, pages 68–75, 2009.

[9] Franck Le, Geoffrey G. Xie, and Hui Zhang. Theory and new primitives for safely connecting routing protocol instances. In *Proceedings of the ACM SIGCOMM*, pages 219–230, 2010.

[10] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *Proceeding of the ACM STOC*, pages 57–66, 2008.

[11] T. Luo and K.S. Trivedi. An improved algorithm for coherent-system reliability. *IEEE Transactions on Reliability*, 7(1):73–78, 1998.

[12] Y. Ohara. *Routing Architecture for the Dependable Internet*. PhD thesis, Keio University, 2008.

[13] S. Rai, M. Veeraraghavan, and K. S. Trivedi. A survey of efficient reliability computation using disjoint products approach. *Networks*, 25(3):147–163, 1995.

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

[15] R. Sami, M. Schapira, and A. Zohar. Searching for stability in interdomain routing. In *Proceeding of the IEEE INFOCOM*, pages 549–557, 2009.

[16] J. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking*, 13(5):1160–1173, 2003.