

# Routing Betweenness Centrality

SHLOMI DOLEV, YUVAL ELOVICI, AND RAMI PUZIS

*Ben-Gurion University of the Negev, Beer-Sheva, Israel*

**Abstract.** Betweenness-Centrality measure is often used in social and computer communication networks to estimate the potential monitoring and control capabilities a vertex may have on data flowing in the network. In this article, we define the Routing Betweenness Centrality (RBC) measure that generalizes previously well known Betweenness measures such as the Shortest Path Betweenness, Flow Betweenness, and Traffic Load Centrality by considering network flows created by arbitrary loop-free routing strategies.

We present algorithms for computing RBC of all the individual vertices in the network and algorithms for computing the RBC of a given group of vertices, where the RBC of a group of vertices represents their potential to collaboratively monitor and control data flows in the network. Two types of collaborations are considered: (i) conjunctive—the group is a sequence of vertices controlling traffic where all members of the sequence process the traffic in the order defined by the sequence and (ii) disjunctive—the group is a set of vertices controlling traffic where at least one member of the set processes the traffic. The algorithms presented in this paper also take into consideration different sampling rates of network monitors, accommodate arbitrary communication patterns between the vertices (traffic matrices), and can be applied to groups consisting of vertices and/or edges.

For the cases of routing strategies that depend on both the source and the target of the message, we present algorithms with time complexity of  $O(n^2m)$  where  $n$  is the number of vertices in the network and  $m$  is the number of edges in the routing tree (or the routing directed acyclic graph (DAG) for the cases of multi-path routing strategies). The time complexity can be reduced by an order of  $n$  if we assume that the routing decisions depend solely on the target of the messages.

Finally, we show that a preprocessing of  $O(n^2m)$  time, supports computations of RBC of sequences in  $O(kn)$  time and computations of RBC of sets in  $O(k^3n)$  time, where  $k$  is the number of vertices in the sequence or the set.

Categories and Subject Descriptors: C.2.3 [Computer Communication Networks]: Network Operations—*Network monitoring*; F.2.m [Analysis of Algorithms and Problem Complexity]: Numerical

25

---

This research was partially supported by Deutsche Telekom AG, the Rita Altura Trust Chair in Computer Sciences, and the ICT Programme of the European Union under contract number FP7-215270 (FRONTS).

Authors' addresses: S. Dolev, Department of Computer Science, Ben-Gurion University of the Negev and Deutsche Telekom Laboratories at Ben-Gurion University of the Negev, Beer-Sheva, 84105, Israel, e-mail: dolev@cs.bgu.ac.il; Y. Elovici, Deutsche Telekom Laboratories at Ben-Gurion University of the Negev and the Department of Information Systems Engineering, Beer-Sheva, 84105, Israel, e-mail: elovici@inter.net.il. R. Puzis, Deutsche Telekom Laboratories at Ben-Gurion University of the Negev and the Department of Information Systems Engineering, Beer-Sheva, 84105, Israel, e-mail: puzis@bgu.ac.il (Contact Author).

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2010 ACM 0004-5411/2010/04-ART25 \$10.00

DOI 10.1145/1734213.1734219 <http://doi.acm.org/10.1145/1734213.1734219>

Algorithms and Problems—*Miscellaneous*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms, network problems*

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Betweenness, centrality, communication networks, probabilistic routing

**ACM Reference Format:**

Dolev, S., Elovici, Y., and Puzis, R. 2010. Routing betweenness centrality. *J. ACM* 57, 4, Article 25, (April 2010), 27 pages.

DOI = 10.1145/1734213.1734219 <http://doi.acm.org/10.1145/1734213.1734219>

## 1. Introduction

Networks are commonly used to represent a domain, a problem, or a complex dynamic system in a large variety of scopes [Strogatz 2001]. Representing, for example, social networks [Wasserman and Faust 1994], protein interactions [Bork et al. 2004], urban structure [Porta et al. 2006], and computer communication networks [Faloutsos et al. 1999; Yook et al. 2002]. Various centrality measures such as Degree, Closeness, and Betweenness [Freeman 1979; Borgatti and Everett 2006] are used to analyze networks and understand both the global dynamics of the networks and the roles played by individual vertices.

Many naturally evolved complex networks are characterized by a power-law distribution of Degree centrality of their vertices. Barabasi and Albert [1999] referred to such networks as scale-free. For example, the Internet topology [Faloutsos et al. 1999] and the world-wide-web [Barabasi et al. 2000] can be regarded as scale-free networks. Bollobas and Riordan [2003] have shown that these networks are highly resistant to random damages but are easily partitioned by removing the most central vertices. Barthélemy [2004] has shown that many scale-free networks are also characterized by power-law distribution of Betweenness-Centrality. This implies, for example, that most of the communication flows in scale-free networks can be inspected by a small group of collaborating vertices [Puzis et al. 2009].

The centrality characteristics of vertices are also important in the field of computational epidemiology, where Pastor-Satorras and Vespignani [2002] have shown that immunizing central vertices can significantly reduce the impact of epidemics. In the scope of the Internet, Jackson et al. [2007] suggest placing monitors on links of the autonomous system level topology of the Internet, with end vertices having the highest Degree.

In this article, we concentrate on derivations of Betweenness-Centrality, that was defined by Freeman [1977] to estimate the control an individual may have over communication flows in social networks. Betweenness-Centrality measures may be used to estimate the monitoring capabilities, control capabilities, and/or functionality importance of vertices in communication networks. The concept of Betweenness evolved into a broad class of diverse measures that consider different types of network flows [Borgatti 2005]. Betweenness, which is now referred to as Shortest Path Betweenness-Centrality (SPBC) assumes that only shortest paths are used to transfer the network flow. Load Centrality (LC) as defined by Goh et al. [2001] is a variant of betweenness that also assumes that traffic flows over shortest paths, but uses a different routing mechanism. When devising a routing strategy in a commercial communication network, factors such as load balancing, fault tolerance, and service level agreements must be considered. Unfortunately,

these factors may lead to traffic flows that are not routed along shortest paths to the target and, therefore, ignored by SPBC and LC.

Flow Betweenness-Centrality (FBC) proposed by Freeman et al. [1991] equally considers routes of all lengths and assumes that routes are simple (containing no cycles). While simple routes is a reasonable assumption for communication networks, routing strategies in computer network usually do prefer shorter paths over longer paths. For example, in Random Walk Betweenness-Centrality (RWBC), proposed by Newman [2005], shorter paths tend to be accounted for more than longer ones. However, RWBC assumes that routes may contain cycles, which is not the case in most communication networks. Besides the path length issue, each one of the above Betweenness-Centrality measures assumes a fixed communication model that does not fully match routing strategies used in communication networks such as the Internet. In this article, we propose a more flexible and realistic measure called Routing Betweenness Centrality, that accommodates a wide class of routing strategies.

Most routing protocols create routing tables that match the destination address of a packet with one output port. Occasionally, routing tables are changed if one of the links, attached to a router, is unavailable due to malfunctions or congestions. During the time period when routing tables do not change they create spanning trees rooted at every target vertex in the network. Some routing protocols maintain shortest paths to the target while others balance the traffic load on the network by forwarding superfluous traffic though less loaded routes that are not necessarily shortest [Moy 1998; Villamizar 2002]. Routing protocols may utilize multiple paths from source to target which are not necessarily shortest, but it is important to note that, in the stable state they do not contain loops.

Routing Betweenness-Centrality (RBC), as defined in this article accommodates arbitrary loop-free routing schemes where routing decisions depend on the packet target alone or on both the source and the target of the packet. It is easy to show that for computing SPBC, LC, and FBC are particular cases of RBC. We elaborate on SPBC, LC, and FBC in Section 2 and show how to define a routing strategy that will match their communication model in Section 3.

RBC is a generalization of other Betweenness Centrality measures, not only from the point of view of data routing. RBC can be computed with respect to arbitrary overlay networks (a.k.a traffic matrices) similarly to the algorithm for SPBC computation described by Puzis et al. [2007c]. RBC can also be used to simulate many variants of SPBC [Brandes 2008] by means of a specially crafted routing function and a traffic matrix.

In this article, we also add the notion of a sampling rate that was not considered by prior algorithms for computing the variants of Betweenness Centrality.

We present a set of algorithms for computing RBC of individual vertices, sequences of vertices (e.g. links), and sets of vertices and/or links. Table I summarizes the algorithms discussed in this paper. In Section 4, we present algorithms that, given a loop-free routing scheme, compute RBC by topologically sorting all vertices between each source-target pair. In Section 5, we reduce the complexity of RBC computations for routing schemes where the routing decisions are affected only by the target of the packet and how to efficiently compute RBC of sets consisting of both links and vertices. Conclusions appear in Section 6. Symbols and notation principles used in this article are summarized in the Appendix.

TABLE I. TIME AND SPACE COMPLEXITY OF THE PROPOSED ALGORITHMS

Section / Alg.	Scope	Routing depends on	Space	Preproc. time	Query time
4.1 / 1	All vertices	source and target	$O(m)$		$O(n^2m)$
4.2 / 2	Sequence	source and target	$O(m)$		$O(n^2m)$
4.3 / 3	Set (vertices)	source and target	$O(m)$		$O(n^2m)$
5.1 / 4	All vertices	only target	$O(m)$		$O(nm)$
5.2 / 5	Sequence	only target	$O(m)$		$O(nm)$
5.2 / 6	Sequence	only target	$O(n^3)$	$O(n^2m)$	$O(nk)$
5.3 / 7	Set (vertices)	only target	$O(m)$		$O(nm)$
5.3 / 8	Set (vertices)	only target	$O(n^3)$	$O(n^2m)$	$O(k^3n)$
5.4 / 9	Set (links)	only target	$O(n^3)$	$O(n^2m)$	$O(k^3n)$
5.4 / 10	Set (mixed)	only target	$O(n^3)$	$O(n^2m)$	$O(k^3n)$

$n$  – number of vertices in the network;  $m$  – maximal number of edges in routing trees (or a routing directed acyclic graphs (DAG) for multi-path routing schemes);  $k$  – number of vertices in a sequence or a set.

## 2. Preliminaries on Betweenness-Centrality

Shortest Path Betweenness Centrality (SPBC) was introduced in social sciences to measure the potential influence of an individual over the information flow in a social network [Anthonisse 1971; Freeman 1977]. SPBC is defined as the sum of fractions of all shortest paths between each pair of vertices in a network that traverse a given node:

$$SPBC(v) = \sum_{s \neq v \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}},$$

where  $\sigma_{s,t}$  is the number of shortest paths connecting  $s$  and  $t$  and  $\sigma_{s,t}(v)$  is the number of paths between  $s$  and  $t$  that traverse  $v$ . Assume, for example, that the network uses a shortest-path routing scheme where the route is randomly chosen out of all shortest paths from source to target. Assume also that every vertex sends one packet to every other node. In this case, SPBC of a vertex  $v$  is the expected number of packets that traverse  $v$ . Brandes [2001] has shown that shortest paths from a single source  $s$  to all other vertices can be efficiently aggregated by traversing the vertices in the order of a nonincreasing distance from  $s$ . Efficient aggregation of shortest paths is used to compute SPBC of all vertices in a network in  $O(|V||E|)$  time, where  $V$  is the set of vertices and  $|E|$  is the set of links in the network. Everett and Borgatti [1999] have shown that SPBC can naturally be extended to Group Betweenness Centrality (GBC). Given a set of vertices ( $M$ ), GBC is defined as the sum of fractions of all shortest paths which traverse at least one vertex in the set. GBC of a single set can be computed in  $O(|V||E|)$  [Yagil 2005; Brandes 2008] or in  $O(|M|^3)$  time following a preprocessing that takes  $O(|V|^3)$  time [Puzis et al. 2007a].

The definition of GBC resembles the definition of the effectiveness of a group of distributed network monitors which is defined by Cantieni et al. [2006] as the probability that a random packet is sampled at least once by at least one of the monitors. Moreover, Holme [2003] has shown that the SPBC of a vertex is highly correlated with the fraction of time that the vertex is occupied by traffic. SPBC was also used by Yan et al. [2006] for predicting and avoiding congestions. These findings indicate that SPBC can be used as a heuristic in many network related tasks such as designing routing protocols, optimizing deployment of network monitors, finding bottlenecks in the network, etc. Unfortunately, in practice, not all the

shortest paths between source and target have the same probability to transfer a packet as assumed by SPBC.

Traffic Load Centrality (LC) assumes a more realistic routing strategy, where every vertex forewords the packet to a neighbor chosen randomly out of the neighbors that are closest to the target. like SPBC, LC can be computed in  $O(|V||E|)$  time for all vertices in the network [Newman 2001]. The group variant of LC can also be computed in  $O(|V||E|)$  as was indicated by Brandes [2008]. To the best of our knowledge, there are no algorithms that reduce the time required to compute LC of a group of vertices using preprocessing. Zhou et al. [2006] have shown that, although, SPBC and LC possess similar statistical properties, normalized SPBC and LC can differ up to 30% for individual vertices in large networks. The drawback of SPBC and LC measures is that they are both limited to shortest-path routes while in practice traffic flows may deviate from the shortest paths to increase the network performance.

There are some Betweenness-Centrality measures that are not limited to shortest paths. Freeman et al. [1991] introduced Maximal-Flow Betweenness-Centrality (FBC) that equally considers all paths from source to target. Roughly speaking, FBC of the vertex  $v$  is the sum of fractions of the maximal flows between each pair of vertices, that is transferred by the vertex  $v$ :

$$FBC(v) = \sum_{s \neq v \neq t} \frac{\phi_{s,t}(v)}{\phi_{s,t}},$$

where  $\phi_{s,t}$  is maximal flow between  $s$  and  $t$  and  $\phi_{s,t}(v)$  is the portion of this flow that is transferred by the vertex  $v$ . Since the maximal flow can utilize different routes from  $s$  to  $t$ ,  $\phi_{s,t}(v)$  should be averaged over all the possibilities. The main drawback of FBC when applied to communication networks is that it does not prioritize routes according to their lengths, while in practice, most of the traffic is routed through shortest paths.

The work of Borgatti and Everett [2006] categorizes Betweenness-Centrality measures according to the types of routes assumed and provides valuable insights into the formulation and computation of generic Betweenness-Centrality measures. Routing Betweenness-Centrality (RBC) defined in this paper is a generalization of SPBC, FBC, and LC. We present algorithms for computing RBC of individual vertices as well as sets and sequences of vertices. Algorithms presented in this paper are applicable to a general case of loop-free routing schemes where the routing decisions depend on both the source and the target of a packet, and to source-oblivious schemes. For the latter routing schemes, we show that RBC can be computed with the same time complexity as SPBC and LC (namely in  $O(|V||E|)$  time). We also show how these times can be reduced using preprocessing when the size of the evaluated group is small compared to the size of the network.

### 3. Routing Scheme Representation

Throughout this article, we assume a loop-free routing scheme. We ignore temporary loops created by routing oscillations and treat routing oscillations as an unavoidable noise in the system. Instead, we are interested in a superposition of all stable state routing tables. Each routing decision made along a network path is dictated by the network topology and the status of the network. Link failures

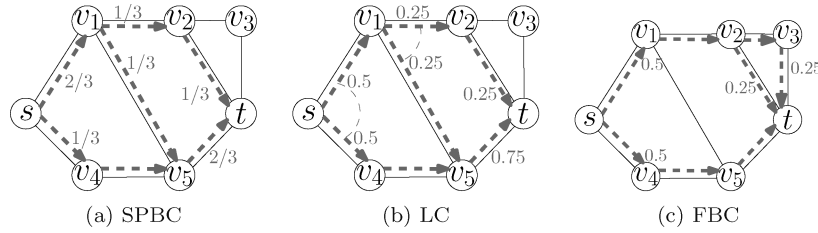


FIG. 1. Sample network with traffic flowing from  $s$  to  $t$  according to SPBC, LC, and FBC flow models.

and congestions cause routing decisions across the network to change from time to time. We assume that either the routing decisions are deterministic or the probabilities for specific routing decisions can be determined (e.g., by analyzing historical behavior of the network).

Formally, let  $G = (V, E)$  be a communication network topology where  $V$  is a set of  $n$  vertices and  $E$  is a set of links between the vertices. We do not allow self loops such as  $(v, v) \in E$ . For the purpose of discussion, we also assume the existence of at most one link between every two vertices. The algorithms described in this paper implicitly support multiple edges between the same vertices due to the nature of the  $R$  function defined here below. Let  $T$  be a traffic matrix where  $T(s, t)$  is the number of packets sent from a source vertex  $s$  to a target vertex  $t$ . In general,  $T(s, t)$  can represent any quantity of interest such as the number of bytes, number of sessions, or the importance of communication between  $s$  and  $t$ .

Assume, for example, that a group of monitors is installed on vertices in a network. The total number of bytes or the total importance of the communication passing through this group can be regarded as its monitoring potential. However, the actual volume of information being monitored depends on the sampling rates of the monitors ( $0 \leq \rho_v \leq 1$ ). Our goal is to compute the total expected number of packets sampled by groups of collaborating monitors. We distinguish between two types of groups: sequences—packets should be sampled by all the members in the order defined by the sequence—and sets—packets should be sampled by at least one member.

Let  $R(s, u, v, t) = p$  be a quaternary function representing the averaged routing scheme where  $p$  is the probability that  $u$  will forward to  $v$  a packet with source address  $s$  and target address  $t$ . Note that we assume that all routing decisions (such as  $(s, u, v, t)$ ) are independent. We will use “don’t care”  $\oslash$  to indicate any value. For example,  $R(\oslash, u, v, \oslash) = 0$  if there is no link from  $u$  to  $v$  and  $R(\oslash, v, v, \oslash) = 1$  by convention.  $R$  defines a directed acyclic graph *DAG* for each source-target pair. Complexity of the algorithms described in this paper depends on the number of links in these DAGs. We denote the maximal number of links in all routing DAGs relevant to the network as  $m$ .

The routing scheme  $R$  can represent various policies of message or flow transfer methods. We can embed in  $R$  some message transfer methods assumed by different Betweenness-Centrality measures. In these cases RBC will produce the same values as would be produced by the original Betweenness-Centrality measure. We will use Figure 1 as a sample network for the following examples.

**LC.** vertices forward packets to one of the neighbors which are closest to the target with equal probability. In this case  $R(s, u, v, t)$  is equal to one divided

by the number of  $v$ 's neighbors that are closest to  $t$ . For example in Figure 1  $R(s, v_1, v_2, t) = 0.5$ .

*SPBC*. vertices forward packets to one of the neighbors which are closest to the target. The probability of  $u$  to forward to  $v$  a packet targeted at  $t$  is equal to the fraction of shortest paths from  $u$  to  $t$  that pass through  $v$   $R(s, u, v, t) = \frac{\sigma_{u,t}(v)}{\sigma_{u,t}}$ . For example in Figure 1  $R(s, s, v_1, t) = 2/3$  since there are three shortest paths between  $s$  and  $t$ , two of which pass through  $v_1$ .

*FBC*. For each  $s, t$  pair vertices forward packets from  $s$  to one of their neighbors to produce maximal flow between  $s$  and  $t$ . The probability of  $u$  to forward to  $v$  a packet from  $s$  to  $t$  is proportional to the portion of the  $s$ - $t$ -flow carried by the undirected link  $(u, v)$ :  $R(s, u, v, t) = \frac{\phi_{s,t}((u,v))}{\phi_{s,t}(u)}$ . For example, if we assume that in Figure 1 the capacity of all links is 0.5, then  $R(s, v_1, v_2, t) = 0.5$  since the link  $(v_1, v_5)$  is not utilized by the maximal flow between  $s$  and  $t$ .

#### 4. Routing Betweenness-Centrality

In this section, we define Routing Betweenness-Centrality (RBC), focusing on routing schemes where the routing decisions depend on the source and the target of a packet. In Section 5 we will show how the computation of RBC can be optimized when the routing decisions are source-oblivious. In the next three subsections we present algorithms for computing RBC of individual vertices, sequences of vertices, and sets of vertices.

**4.1. RBC OF INDIVIDUAL VERTICES.** Assume that a packet is introduced to the network by source vertex  $s$  and destined to leave the network at target vertex  $t$ . Let  $\delta_{s,t}(v)$  be the probability that this packet will pass through the vertex  $v$ . We will refer to  $\delta_{s,t}(v)$  and its variants as *pairwise dependency* of  $s$  and  $t$  on the intermediate  $v$ .  $\delta_{s,t}(v) \cdot T(s, t)$  is the expected number of packets sent from  $s$  to  $t$  that pass through  $v$ . Note that for special cases where  $v$  equals  $s$ ,  $t$ , or both it holds that  $\delta_{s,t}(s) = \delta_{s,t}(t) = 1$ .  $\delta_{s,t}(v)$  can be recursively computed for arbitrary  $v \in V$  based on the loop-free routing strategy  $R(s, u, v, t)$ . Let  $Pred_{s,t}(v)$  be a set of all immediate predecessors of  $v$  on the way to  $t$ :  $Pred_{s,t}(v) = \{u | R(s, u, v, t) > 0\}$ . Let  $u$  be a predecessor of  $v$  on the way from  $s$  to  $t$ . The probability that a packet will pass through  $v$  after visiting  $u$  is  $R(s, u, v, t)$ . Hence, the pairwise dependency of  $s$  and  $t$  on  $v$  can be computed using pairwise dependency of  $s$  and  $t$  on  $v$ 's predecessors.

$$\begin{aligned} \delta_{s,t}(s) &= 1 \\ \delta_{s,t}(v) &= \sum_{u \in Pred_{s,t}(v)} \delta_{s,t}(u) \cdot R(s, u, v, t). \end{aligned} \tag{1}$$

Since we assume loop-free routing,  $Pred_{s,t}$  defines a directed acyclic graph (DAG) [Harary et al. 1965] as shown in Figure 2. Therefore, we can compute  $\delta_{s,t}(v)$  for all  $v \in V$  in  $O(m)$  in the worst case. All we need to do is topologically sort the DAG induced by  $Pred_{s,t}$  and iteratively apply Eq. (1) on all vertices starting from  $s$ .

Let RBC of a vertex  $v$  ( $\delta_{\bullet,\bullet}(v)$ ) be the expected number of packets that pass through  $v$ .

$$\delta_{\bullet,\bullet}(v) = \sum_{s,t \in V} \delta_{s,t}(v) \cdot T(s, t). \tag{2}$$

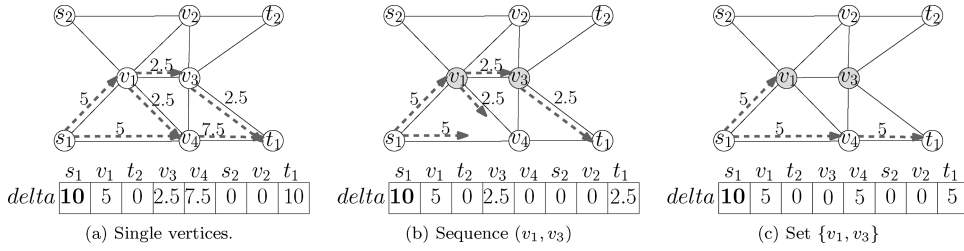


FIG. 2. Example of a routing DAG from  $s_1$  to  $t_1$  (dashed gray arrows). In this example we assume  $T(s_1, t_1) = 10$ . The numbers on the arrows in sub-figures (a), (b), and (c) indicate the  $\delta$  values contributed by the topologically sorted vertices to their successors in Algorithms 1, 2, and 3 respectively.

$\delta_{\bullet,\bullet}(v)$  can be regarded as the potential of  $v$  to inspect or alter communications in the network. Eq. (2) resembles the original definition of SPBC with two exceptions. First, each  $\delta_{s,t}(v)$  is multiplied by the number of packets sent from  $s$  to  $t$  to compute the traffic load on  $v$ . Second, end points are included in the summation to accommodate communications originating from (or destined to) the investigated node. Algorithm 1 computes the RBC of all individual vertices in  $O(n^2m)$  time using Eqs. (1) and (2).

Algorithm 1 is composed of an outer loop that iterates over all  $s$ - $t$  pairs of vertices and of three inner stages. In the first stage the algorithm creates the routing DAG with single source  $s$  and single sink  $t$ . In the second stage the  $\delta$  array is initialized (bold number in Figure 2). Entry  $\delta[v]$  of this array represents the expected number of packets from  $s$  to  $t$  that pass through  $v$ :  $\delta_{s,t}(v) \cdot T(s, t)$ . Finally in the third stage the expected number of packets from  $s$  to  $t$  that pass through

---

**Algorithm 1.** RBC of vertices

---

**Input:**  $G(V, E)$ ,  $R$ ,  $T$

**Output:**  $RBC[1..|V|]$

**Data:**  $\delta[1..|V|]$

$\forall v \in V, RBC[v] = 0$

**For**  $s, t \in V$  **do**

  ▼ **topological sort**

$E' = \{(u, v) | R(s, u, v, t) > 0\}$   
     $D =$  directed acyclic graph  $(V, E')$   
     $\{s = v_0 \leq v_1 \leq \dots \leq v_n = t\} \leftarrow$   
    topologically sorted vertices of  $D$

  ▼ **init delta**

$\forall v \in V, \delta[v] = 0;$   
     $\delta[s] = T(s, t)$

  ▼ **accumulate  $\delta_{\bullet,\bullet}(v)$**

**for**  $i = 0$  **to**  $n$  **do**  
      **for**  $v_j \in \text{successors}(v_i)$  **do**  
         $\delta[v_j] +=$   
         $\delta[v_i] \cdot R(s, v_i, v_j, t)$   
    **for**  $v \in V$  **do**  
       $RBC[v] += \delta[v]$

**return**  $RBC$

---



each one of the vertices is computed and these probabilities are accumulated according to Equation 2 to form RBC values of all vertices. Most of the following algorithms will use the same template and similar content.

**4.2. RBC OF ORDERED SEQUENCES.** In this section, we define RBC of ordered sequences of vertices. A link is a private case of a sequence of size two where the members of the sequence are connected. Betweenness-Centrality of a sequence measures the extent to which packets traverse all the vertices in the sequence in a given order. For example, RBC of a sequence of monitors can reveal the level of redundant traffic inspection. The SPBC of sequences was first mentioned by Puzis et al. [2007a] as a technique to speed up the computation of GBC. We will also use the concept of sequence Betweenness-Centrality to speed up the computation of RBC of sets in Section 5.3.

Let  $S = (s_1, \dots, s_k)$  be a sequence of vertices. Let  $\tilde{\delta}_{s,t}(S)$  be the probability that a single packet emanating from  $s$  and targeted at  $t$  will pass through all vertices in the sequence  $S$ , first through  $s_1$  then through  $s_2$  and so on until  $s_k$ .  $\tilde{\delta}_{s,t}(S) \cdot T(s, t)$  is the expected number of packets sent from  $s$  to  $t$  that pass through  $S$ . The sequence  $S$  can be any finite sequence of vertices. If the same vertex appears more than once, all successive appearances of the vertex can be reduced to one instance, for example  $\tilde{\delta}_{s,t}((u, v, v, v, w)) = \tilde{\delta}_{s,t}((u, v, w))$ . On the other hand, if two appearances of a vertex in the sequence  $S$  are separated by a different vertex this will create a cycle and  $\tilde{\delta}(S)$  will be equal to zero according to the assumption of loop-free routing. For the same reason,  $\tilde{\delta}_{s,t}(S)$  is equal to zero if  $S$  contains  $s$  following some other vertices, for example  $\tilde{\delta}_{s,t}((v, \dots, s, \dots)) = 0$ . The following set of equations recursively computes the probability that a packet will pass through the sequence  $S$ :

$$\begin{aligned} \tilde{\delta}_{s,t}((s)) &= 1 \\ (v_{k-1} = v_k) \quad \tilde{\delta}_{s,t}((\dots, v_{k-1}, v_k)) &= \tilde{\delta}_{s,t}((\dots, v_{k-1})) \\ (v_{k-1} \neq v_k) \quad \tilde{\delta}_{s,t}((\dots, v_{k-1}, v_k)) &= \sum_{u \in \text{Pred}_{s,t}(v_k)} \tilde{\delta}_{s,t}((\dots, v_{k-1}, u)) \cdot R(s, u, v_k, t). \end{aligned} \quad (3)$$

The set of predecessors ( $\text{Pred}_{s,t}(r)$ ) remains the same as in previous subsection. Therefore, the Eq. (3) can also be solved in  $O(m)$  time similarly to Eq. (1).

Let  $S_\rho = (s_1, \dots, s_k)$  be a sequence of vertices with sampling rates  $\rho_{s_1}, \dots, \rho_{s_k}$  respectively. For simplicity of the following discussion we assume that all vertices in  $S$  are different. We will denote by  $S$  the same sequence of vertices disregarding their sampling rates. The probability that a packet from  $s$  to  $t$  will be sampled by all vertices in  $S_\rho$  is the probability that it will pass through  $S$  multiplied by the product of sampling rates of all vertices in the sequence. RBC of an ordered sequence of vertices  $S_\rho$  (denoted by  $\tilde{\delta}_{\bullet,\bullet}(S_\rho)$ ) is defined as the expected number of packets sampled by all vertices in  $S_\rho$  in a given order.

$$\tilde{\delta}_{\bullet,\bullet}(S_\rho) = \prod_{r \in S} \rho_r \cdot \sum_{s,t \in V} \tilde{\delta}_{s,t}(S) \cdot T(s, t) \quad (4)$$

**Algorithm 2.** RBC of sequences (with sampling)

---

**Input:**  $G(V, E), R, T, \rho, S = (s_0, s_1, \dots, s_l)$   
 $(i \neq j \Rightarrow s_i \neq s_j)$   
**Output:** RBC of  $S$   
**Data:**  $\text{delta}[1..|V|]$   
 $\text{RBCofS} = 0$   
**for**  $s, t \in V$  **do**  
  ▶ **topological sort**  
  ▶ **init delta**  
  ▼ **accumulate**  $\ddot{\delta}_{\bullet, \bullet}(S_\rho)$   
     $k = 0$   
    **for**  $i = 0$  **to**  $n$  **do**  
      **if**  $v_i = s_k$  **then**  $k++ = 1$   
      **for**  $v_j \in \text{successors}(v_i)$  **do**  
        **if**  $v_j < s_k$  **or**  $v_j$  *is*  $s_k$  **then**  
           $\text{delta}[v_j] +=$   
           $\text{delta}[v_i] \cdot R(s, v_i, v_j, t);$   
     $\text{RBCofS} += \text{delta}[t] \cdot \prod_{s_i \in S} (\rho_{s_i});$   
**return**  $\text{RBCofS}$

---

Note that RBC of a directed link  $(u, v) \in E$  and a single vertex  $w \in V$  is simply RBC of the sequences  $(u, v)$  and  $(w)$ , respectively. Eqs. (3) and (4) can be used to compute RBC of one sequence of vertices in  $O(n^2m)$ .  $\ddot{\delta}_{\bullet, \bullet}(S_\rho)$  is computed by Algorithm 2, by propagating only the portion of traffic that was sampled by the monitors in  $S_\rho$ .

**4.3. RBC OF SETS.** In this section, we define the set variant of RBC. Generally, Betweenness-Centrality of a group of vertices measures the extent to which packets traverse at least one of the vertices in the group. The concept of centrality was first applied to groups and classes of vertices in networks by Everett and Borgatti [1999]. The set variant of RBC can be used, for example, for estimating the expected effectiveness of distributed monitors.

Let  $M = \{v_0, \dots, v_k\}$  be a set of vertices. Let  $\ddot{\delta}_{s,t}(M)$  be the probability that a packet from  $s$  to  $t$  will pass through at least one of the vertices in  $M$ .  $\ddot{\delta}_{s,t}(M) \cdot T(s, t)$  is the expected number of packets sent from  $s$  to  $t$  that pass through  $M$ . If we disregard sampling rates, RBC of set  $M$  is:

$$\ddot{\delta}_{\bullet, \bullet}(M) = \sum_{s, t \in V} \ddot{\delta}_{s,t}(M) \cdot T(s, t).$$

Let  $\rho_v$  be the sampling rate of the monitor installed on the vertex  $v$ . Let  $M_\rho = \{v | \rho_v > 0\}$  be a set of vertices with positive sampling rates.  $M_\rho$  can be regarded as a fuzzy set where  $\rho_v$  is the extent to which  $v$  belongs to  $M_\rho$ . Let  $\ddot{\delta}_{s,t}(M_\rho)$  be the probability that a packet from  $s$  to  $t$  will be sampled by at least one of the vertices in  $M$ . For the sake of simplicity, we prefer to compute  $\ddot{\delta}_{s,t}(M_\rho)$  using its inverse probability, namely the probability that a packet from  $s$  to  $t$  will not be sampled by monitors in  $M$ . Assume, for example, that each sampled packet is marked by the monitors. Let  $\lambda_{s,t}^{M_\rho}(v)$  be the probability that a packet from  $s$  to  $t$  will pass through  $v$  without being marked neither before arriving to  $v$  nor by  $v$  itself. The probability that a packet from  $s$  to  $t$  will not be marked by  $v$  is  $1 - \rho_v$ . Therefore, the probability that the packet will leave  $s$  without being marked is  $1 - \rho_s$ . Let  $u$  be a predecessor

**Algorithm 3.** RBC of sets (with sampling)**Input:**  $G(V, E)$ ,  $R$ ,  $T$ ,  $\rho$ **Output:**  $RBC$ **Data:**  $\delta_{\bullet,\bullet}[1..|V|]$ , totalTraffic $RBC = 0$ **for**  $s, t \in V$  **do**    **►** topological sort    **►** init delta    **▼** accumulate  $\delta_{\bullet,\bullet}(M_\rho)$         totalTraffic =  $\sum_{i=0}^{n-1} \delta_{\bullet,\bullet}[i]$         **for**  $i = 0$  **to**  $n$  **do**             $\delta_{\bullet,\bullet}[v_i] = \delta_{\bullet,\bullet}[v_i] \cdot (1 - \rho_{v_i})$             **for**  $v_j \in \text{successors}(v_i)$  **do**                 $\delta_{\bullet,\bullet}[v_j] +=$   
                     $\delta_{\bullet,\bullet}[v_i] \cdot R(s, v_i, v_j, t);$          $RBC += (\text{totalTraffic} - \delta_{\bullet,\bullet}[t]);$ **return**  $RBC$ 

of  $v$ . A product  $\lambda_{s,t}^{M_\rho}(u) \cdot R(s, u, v, t)$  is the probability that the packet will reach  $v$  through  $u$  without being marked. Summing these products over all predecessors of  $v$  will result in the probability that the packet will get to  $v$  without being marked as shown in Eq. (5).

$$\lambda_{s,t}^{M_\rho}(v) = (1 - \rho_v) \cdot \sum_{u \in \text{Pred}_{s,t}(v)} \lambda_{s,t}^{M_\rho}(u) \cdot R(s, u, v, t) \quad (5)$$

$\lambda_{s,t}^{M_\rho}(t)$  is the probability that the packet from  $s$  to  $t$  will not be sampled by any of the monitors. Therefore,  $(1 - \lambda_{s,t}^{M_\rho}(t)) \cdot T(s, t)$  is the expected number of distinct packets from  $s$  to  $t$  captured by the monitors:

$$\delta_{s,t}(M_\rho) = 1 - \lambda_{s,t}^{M_\rho}(t).$$

The RBC of the fuzzy set  $M_\rho$  is the expected number of packets sampled by at least one vertex in  $M_\rho$  and can be computed using the inverse probabilities as described in Eq. (6),

$$\delta_{\bullet,\bullet}(M_\rho) = \sum_{s,t \in V} (1 - \lambda_{s,t}^{M_\rho}(t)) \cdot T(s, t) \quad (6)$$

Assume a vertex  $v \in V$  and sampling rates  $\rho$  such that  $\rho_v = 1$  and for each  $u \neq v$ ,  $\rho_u = 0$ . In this case,  $\delta_{\bullet,\bullet}(v) = \delta_{\bullet,\bullet}(\{v\}_\rho)$  making RBC of sets a valid generalization of RBC of single vertices. In the following discussions, we will occasionally omit the subscript  $\rho$  notation when referring solely to the vertices in  $M$  or when sampling rates are assumed to be 0 or 1.

Eqs. (5) and (6) can be used to compute RBC of one group of monitors with given sampling rates in  $O(n^2m)$  as shown in Algorithm 3. In the input to Algorithm 3, we use  $\rho$  to represent vertices with positive sampling rates. In the propagation stage of Algorithm 3, the only traffic that was not sampled propagates until it reaches  $t$ .

Algorithm 3 is composed of an outer loop with three inner stages similarly to Algorithm 1. The first two phases remain intact. The third phase implements Eq. (5) to fill the *delta* array with the expected number of packets from  $s$  to  $t$  that were not captured before or at the respective node. In addition, instead of computing RBC of all vertices in the networks, the algorithm computes the total expected number of packets that were captured by at least one monitor according to Eq. (6).

This concludes the definition of RBC and its computation methods for routing strategies where the routing decisions depend on both the source and the target of a packet. Next, we will show how the assumption of source-oblivious routing reduces the time complexity of the presented algorithms from  $O(n^2m)$  to  $O(nm)$ .

### 5. Computing RBC for Source-Oblivious Routing

In this section, we will describe how the computation of RBC can be optimized when assuming a source-oblivious routing scheme. We will revise the computation of RBC of single vertices, sets, and sequences and present their respective algorithms with minimal changes.

**5.1. RBC OF INDIVIDUAL VERTICES.** Let  $\delta_{\bullet,t}(r)$  be the expected number of packets targeted at  $t$  that pass through the vertex  $r$  as defined by Eq. (7).

$$\delta_{\bullet,t}(r) = \sum_{s \in V} \delta_{s,t}(r) \cdot T(s, t) \quad (7)$$

$\delta_{\bullet,t}(r)$  estimates the ability of  $r$  to monitor traffic flows targeted at  $t$ . We will refer to  $\delta_{\bullet,t}(r)$  as *target dependency* of  $t$  on  $r$ . In this and following sections, we will show how to compute RBC of individual vertices, sequences, and sets by aggregating target dependencies. Since target dependency is a summation of pairwise dependencies over all sources, RBC of the vertex  $r$  is a summation of target dependencies over all targets as shown in Eq. (8).

$$\delta_{\bullet,\bullet}(r) = \sum_{t \in V} \delta_{\bullet,t}(r) \quad (8)$$

If we are able to compute target dependency directly without using Eq. (7), the computation of  $\delta_{\bullet,\bullet}(r)$  can be accelerated by replacing the loop over all  $s - t$  pairs in Algorithm 1 by a loop over all target vertices  $t$  only. Next, we will show that target dependency can be computed recursively similarly to the computation of pairwise dependency. The similarity between these computations will allow us introducing only minimal changes to the pseudo code of Algorithm 1 in order to adapt it to source-oblivious routing strategies and reduce its complexity.

Let  $Pred_t(v)$  be a set of all predecessors of  $v$  on the way to  $t$ :  $Pred_t(v) = \{u | R(\odot, u, v, t) > 0\}$ . In contrast to  $Pred_{s,t}(v)$  defined in Section 4, here the set of the possible predecessors of  $v$  is not influenced by the source of communication. Let  $u$  be a predecessor of  $v$  on the way to  $t$ . The probability of a packet to pass through  $v$  after visiting  $u$  is  $R(\odot, u, v, t)$ . The expected number of packets targeted at  $t$  that can be monitored by  $v$  include packets introduced to the network by  $v$  ( $T(v, t)$ ) and all packets introduced or forwarded by  $v$ 's predecessors as described

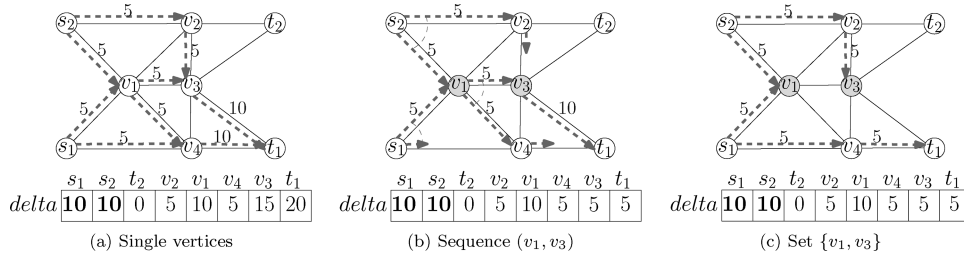


FIG. 3. Example of a source-oblivious routing DAG with a single sink  $t$  and two sources (dashed gray lines). In this example we assume  $T(s_1, t_1) = T(s_2, t_1) = 10$  and  $T(v_i, t_1) = 0$ . The numbers on the arrows in sub-figures (a), (b), and (c) indicate the  $\delta$  values contributed by the topologically sorted vertices to their successors in Algorithms 1, 2, and 3, respectively.

by Eq. (9).

$$\delta_{\bullet,t}(v) = T(v, t) + \sum_{u \in \text{Pred}_t(v)} \delta_{\bullet,t}(u) \cdot R(\emptyset, u, v, t). \quad (9)$$

This equation can be derived directly from Eqs. (1) and (7), which describe the computation of  $\delta_{s,t}(v)$  and define  $\delta_{\bullet,t}(v)$ , respectively.

Since we assume loop-free routing  $\text{Pred}_t$  defines a DAG similarly to  $\text{Pred}_{s,t}$ , but this time the DAG has multiple sources and a single sink  $t$  as shown in Figure 3. Eq. (9) allows computing the values of  $\delta_{\bullet,t}(v)$  for all  $v \in V$  in  $O(m)$ , in the worst case. Structural similarity of Eqs. (1) and (9) suggests that the same process can be used to compute  $\delta_{s,t}(v)$  and  $\delta_{\bullet,t}(v)$ . In fact, by changing the “init delta” stage of Algorithm 1 as shown in Algorithm 4 we make the accumulation stage fill the  $\delta$  array with target dependencies instead of pairwise dependencies. Algorithm 4 initializes each entry of the array  $\delta[v]$  with  $T(v, t)$  instead of assigning  $T(s, t)$  to  $\delta[s]$  and zero to all other entries.

In contrast to Algorithms 1, 2, and 3, that loop through all  $s$ - $t$  pairs of vertices, we need to loop only through all target vertices to compute RBC given the source-oblivious routing strategy. Algorithm 4 loops once through all target vertices  $t \in V$ , performing a three-stage operation similar to Algorithm 1. In the first stage, the algorithm builds the routing DAG with multiple sources and a single sink (opposed

---

**Algorithm 4.**  $s$ -oblivious RBC of vertices)

---

**Input:**  $G(V, E)$ ,  $R$ ,  $T$

**Output:**  $\text{RBC}[1..|V|]$

**Data:**  $\delta[1..|V|]$

$\forall v \in V, \text{RBC}[v] = 0$

**for**  $t \in V$  **do**

  ▼ **topological sort**

$E' = \{(u, v) \mid R(\emptyset, u, v, t) > 0\}$

$D =$  directed acyclic graph  $(V, E')$

$\{v_0 \leq v_1 \leq \dots \leq v_n = t\} \leftarrow$

    topologically sorted vertices of  $D$ ;

  ▼ **init delta**

$\forall v \in V, \delta[v] = T(v, t);$

  ► **accumulate**  $\delta_{\bullet,t}(v)$ ;

**return**  $\text{RBC}$

---

to the single source and single sink DAG, built by the algorithms in the previous section), sorting its vertices. In the second stage, the *delta* array is initialized to  $T(v, t)$ . For example, in Figure 3,  $T(s_1, t_1) = T(s_2, t_1) = 10$ . Finally, in the third stage, the algorithm traverses the topologically sorted vertices of the network and aggregates RBC values. The third stage remains the same as in Algorithm 1, despite the fact that the *delta* array now represents target dependencies and not pairwise dependencies.

Algorithm 4 iterates once over all vertices in the network, and performs for each one of them a computation that takes at most  $O(m)$  steps. Thus, the overall complexity of the algorithm is  $O(nm)$ . This is an order of magnitude faster than Algorithm 1, whose complexity is  $O(n^2m)$ . Next we present the equations which adapt RBC computation of sequences and sets to the semantics of target dependencies.

## 5.2. RBC OF ORDERED SEQUENCES.

*Employing Target Dependency.* Let  $S_\rho = (s_1, \dots, s_k)$  be a sequence of vertices with sampling rates  $\rho_{s_1}, \dots, \rho_{s_k}$  respectively. Let  $\tilde{\delta}_{\bullet,t}(S)$  be the expected number of packets targeted at  $t$  that pass through all vertices in the sequence  $S$ :

$$\tilde{\delta}_{\bullet,t} = \sum_{s \in V} \tilde{\delta}_{s,t}(S) \cdot T(s, t).$$

Accordingly,  $\tilde{\delta}_{\bullet,t}(S) \cdot \prod_{v \in S} (\rho_v)$  is the expected number of packets targeted at  $t$  that are sampled by all vertices in the sequence. Eqs. (10) and (11) describe RBC of the sequence  $S_\rho$  in terms of  $\tilde{\delta}_{\bullet,t}(S_\rho)$ .

$$\tilde{\delta}_{\bullet,t}((v)) = \delta_{\bullet,t}(v) \tag{10}$$

$$(v_{k-1} = v_k) : \tilde{\delta}_{\bullet,t}((\dots, v_{k-1}, v_k)) = \tilde{\delta}_{\bullet,t}((\dots, v))$$

$$(v_{k-1} \neq v_k) : \tilde{\delta}_{\bullet,t}((\dots, v_{k-1}, v_k))$$

$$= \sum_{u \in \text{Pred}_t(v_k)} \tilde{\delta}_{\bullet,t}((\dots, v_{k-1}, u)) \cdot R(\emptyset, u, v_k, t)$$

$$\tilde{\delta}_{\bullet,\bullet}(S_\rho) = \prod_{v \in S} \rho_v \cdot \sum_{t \in V} \tilde{\delta}_{\bullet,t}(S). \tag{11}$$

Algorithm 5 computes RBC of a sequence of monitors in  $O(nm)$  time using Eqs. (10) and (11). During the iteration over all target vertices this algorithm sorts vertices, in the same way as Algorithm 4 and accumulates betweenness, in the same way as Algorithm 3. Entries of the *delta*[ $v$ ] array represent the expected number of packets sampled by all monitors in the sequence preceding  $v$  in the topological order. In particular, all entries *delta*[ $v$ ] preceding the first element in the sequence are initialized to  $T(v, t)$ .

*Using Precomputed Data.* Next, we will closely examine the probability that a packet sent from  $s$  to  $t$  will pass through  $u$  and then through  $v$  ( $\tilde{\delta}_{s,t}((u, v))$ ). Consider Figure 4 as an example. Assume a packet targeted at  $t$  that has reached  $u$ . The probability that this packet will pass through  $v$  on its way to  $t$  does not depend on the source of the packet and on routing decisions made this far. Therefore, we can multiply the probability that the packet from  $s$  to  $t$  will reach  $u$  ( $\delta_{s,t}(u)$ ) by the probability that a packet from  $u$  to  $t$  will reach  $v$  ( $\delta_{u,t}(v)$ ) to get the probability that

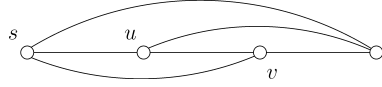


FIG. 4. In this figure, assume that packets are sent from  $s$  to  $t$  and are forwarded by  $u$  and  $v$  from the left to the right. The probability that an arbitrary packet sent from  $s$  to  $t$  will pass through  $u$  and  $v$  is smaller than the probability that it will pass through  $u$ .  $\delta_{s,t}(u) = \frac{1}{3}$ ,  $\delta_{s,t}(v) = \frac{1}{2}$ ,  $\delta_{s,v}(u) = \frac{1}{2}$ , and  $\delta_{u,t}(v) = \frac{1}{2}$ .  $\tilde{\delta}_{s,t}((u, v)) = \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{6}$  since we have two decision points: first on  $s$  and then on  $u$ . Note that  $\frac{1}{6} = \tilde{\delta}_{s,t}((u, v)) \neq \delta_{s,v}(u) \cdot \delta_{s,t}(v) = \frac{1}{4}$ . This is because  $\delta_{s,v}(u)$  does not consider the ultimate target ( $t$ ) and ignores one possible path from  $s$  to  $t$ .

---

**Algorithm 5.**  $s$ -oblivious RBC of sequences (with sampling)

---

**Input:**  $G(V, E)$ ,  $R$ ,  $T$ ,  $\rho$ ,  $S = \{s_0, \dots, s_k\}$

**Output:** RBC

**Data:**  $\text{delta}[1..|V|]$

RBC = 0

**for**  $t \in V$  **do**

    ► **topological sort**

    ▼ **init delta**

**for**  $v \in V$  **do**

**if**  $v < s_0$  **or**  $v$  is  $s_0$  **then**

$\text{delta}[v] = T(v, t)$ ;

**else**

$\text{delta}[v] = 0$ ;

        ► **accumulate**  $\tilde{\delta}_{s,\bullet}(S_\rho)$ ;

**return** RBC

---

a packet from  $s$  to  $t$  will pass through both  $u$  and  $v$ :

$$\tilde{\delta}_{s,t}((u, v)) = \delta_{s,t}(u) \cdot \delta_{u,t}(v).$$

We can add more vertices to the sequence  $(u, v)$  using the following lemma:

**LEMMA 1 (DEPENDENCY CHAINING).** *Let  $S = (s_1, \dots, s_k)$  be an ordered sequence of vertices. The probability that a packet sent from a vertex  $s$  to a different vertex  $t$  will pass through all vertices in  $S$  in a given order is:*

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = \delta_{s,t}(s_1) \cdot \tilde{\delta}_{s_1,t}((s_2, \dots, s_k)).$$

**PROOF.** The following proof is based on the fact that the probability of a packet passing through  $(s_2, \dots, s_k)$ , assuming that the packet already visited  $s_1$ , does not depend on the source of the packet since we assume that the routing scheme under investigation is source-oblivious.

First, we will prove the lemma for  $\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = 0$ .  $\tilde{\delta}_{s,t}((s_1, \dots, s_k))$  is the probability that a packet emanating from  $s$  and targeted to  $t$  will first pass through  $v_1$ , then through  $s_2$ , and so on, until  $s_k$ . This is a non-zero probability if, and only if, there is at least one route from  $s$  to  $t$  that passes through  $(s_1, \dots, s_k)$  in this order. Such a route exists if, and only if, there is a route from  $s$  to  $t$  traversing  $s_1$  and there is a complement route from  $s_1$  to  $t$  that includes the vertices  $s_2, \dots, s_k$ .

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = 0 \Leftrightarrow \delta_{s,t}(s_1) \cdot \tilde{\delta}_{s_1,t}((s_2, \dots, s_k)) = 0.$$

Before we continue the proof for a more general case  $\tilde{\delta}_{s,t}((s_1, \dots, s_k)) \neq 0$ , we will now show that for any set of vertices there is at most one permutation  $L$

of these vertices for which  $\tilde{\delta}_{s,t}(L) \geq 0$ . Note that  $\tilde{\delta}$  is defined as probability and therefore cannot be negative.

**PROPOSITION 1.** *Let  $s, t \in V$  be two vertices in the network. Let  $M \subseteq V$  be a subset of vertices. Let  $L_1$  and  $L_2$  be two permutations of  $M$ . Then for any loop-free routing strategy where routing decisions depend solely on  $s$  and  $t$  (or only  $t$  in case of source-oblivious routing), the following two options are mutually exclusive unless  $L_1 = L_2$ :*

- (1)  $\tilde{\delta}_{s,t}(L_1) > 0$
- (2)  $\tilde{\delta}_{s,t}(L_2) > 0$ .

**PROOF.** Let  $L_1 = (v_1, \dots, v_l)$  and  $L_2 = (u_1, \dots, u_l)$  be two different permutations of  $M$  such that  $\tilde{\delta}_{s,t}(L_1) > 0$  and  $\tilde{\delta}_{s,t}(L_2) > 0$ . Let  $i$  be the lowest integer such that  $v_i \neq u_i$ . Let  $j$  be the index of the vertex  $v_i$  in  $L_2$  ( $v_i = u_j$ ). Let  $k$  be the index of the vertex  $u_i$  in  $L_1$  ( $u_i = v_k$ ). Since all vertices appear only once in both permutations and  $i$  is the lowest index for which vertices are different, it holds that  $i < j$  and  $i < k$ . Without loss of generality, assume that  $j \leq k$ . This means that for each one of the permutations there is at least one route from  $s$  to  $t$  passing through all the vertices in the order defined by the permutation. In particular, it holds that there is at least one route from  $s$  to  $t$  passing through  $(v_i, \dots, v_k)$  and similarly for  $(u_i, \dots, u_j)$ . Since routing decisions depend solely on  $s$  and  $t$  there is a non-zero probability that a packet from  $s$  to  $t$  will reach  $u_j = v_i$  through  $(u_i, \dots, u_j = v_i)$  and continue back to  $u_i = v_k$  through  $(v_i, \dots, v_k = u_i)$  in contradiction to the assumption that the routing strategy is loop-free.  $\square$

Note that the existence of a route from  $s$  to  $t$  through  $(s_1, \dots, s_k)$  implies that there is no route that passes through these vertices in a different order, according to the above Proposition 1. Therefore, if  $\tilde{\delta}_{s,t}((s_1, \dots, s_k)) > 0$ , then the order of vertices  $s, s_1, \dots, s_k, t$  is well defined (with  $s$  being the first node).

Assume that  $\tilde{\delta}_{s,t}((s_1, \dots, s_k)) > 0$ . Let the event  $Z_v$  represent all cases where the packet passes through vertex  $v$ . Let the event  $T_v$  represent all cases where the packet is targeted toward  $v$ . “Targeting” here is different from “passing through” since the target of a packet has an affect on routing decisions along the traversed path.

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = Pr \left[ \bigcap_{i=1}^k Z_{s_i} | Z_s \cap T_t \right] \quad (12)$$

The next equation immediately follows from Eq. (12) since  $\bigcap_{i=1}^k Z_{s_i}$  can be decomposed into two joint events  $Z_{s_1}$  and  $\bigcap_{i=2}^k Z_{s_i}$ .

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = Pr [Z_{s_1} | Z_s \cap T_t] \cdot Pr \left[ \bigcap_{i=2}^k Z_{s_i} | Z_s \cap Z_{s_1} \cap T_t \right] \quad (13)$$

Since we are dealing with source-oblivious routing the vertices that the packet passed prior to passing through  $s_1$  (in particular the source vertex  $s$ ) have no effect on the remaining routing decisions. Therefore,  $s$  has no effect on the probability of



a packet targeted at  $t$  passing through  $s_2, \dots, s_k$  after visiting  $s_1$ .

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = Pr[Z_{s_1} | Z_s \cap T_t] \cdot Pr\left[\bigcap_{i=2}^k Z_{s_i} | Z_{s_1} \cap T_t\right] \quad (14)$$

Finally, in accordance with the definitions of  $\delta$  and  $\tilde{\delta}$ , the proof of Lemma 1 can be completed.

$$\tilde{\delta}_{s,t}((v_0, \dots, v_l)) = \delta_{s,t}(s_1) \cdot \tilde{\delta}_{s_1,t}((s_2, \dots, s_k)) \quad (15)$$

Using Lemma 1, pairwise dependency on a sequence can be represented as a product of pairwise dependencies on single vertices:

$$\tilde{\delta}_{s,t}((s_1, \dots, s_k)) = \delta_{s,t}(s_1) \cdot \delta_{s_1,t}(s_1) \cdot \dots \cdot \delta_{s_{k-1},t}(s_k) \quad (16)$$

Multiplying the Eq. (16) by  $T(s, t)$  and summing it over all sources  $s \in V$  results in a target dependency chain as the following:

$$\tilde{\delta}_{\bullet,t}((s_1, \dots, s_k)) = \delta_{\bullet,t}(s_1) \cdot \prod_{i=1}^k \delta_{v_{i-1},t}(s_i) \quad (17)$$

Eqs. (16) and (17) can be used to compute  $\delta_{s,t}(S)$  and  $\tilde{\delta}_{\bullet,t}(S)$  respectively in  $O(|S|)$  steps given the values of  $\delta_{s,t}(s_i)$  and  $\delta_{\bullet,t}(s_1)$ . Consequently  $\tilde{\delta}_{\bullet,\bullet}(S)$  can be computed in  $O(n \cdot |S|)$  steps using the summation over all target vertices:

$$\tilde{\delta}_{\bullet,\bullet}(S) = \sum_{t \in V} \tilde{\delta}_{\bullet,t}(S).$$

The pseudo-code for the computation can be found in Algorithm 6. The pseudo-code is straight forward and contains two nested loops where the first one iterates over all target vertices in the network. The second loop iterates over the sequence members multiplying the pairwise dependencies.

### 5.3. RBC OF SETS.

*Employing Target Dependency.* Let  $\lambda_{\bullet,t}^{M_\rho}(v)$  be the expected number of packets targeted at  $t$  that reach  $v$  without being captured by any of the vertices in  $M_\rho$ :

$$\lambda_{\bullet,t}^{M_\rho}(v) = \sum_{s \in V} \lambda_{s,t}^{M_\rho}(v) \cdot T(s, t).$$

---

**Algorithm 6.**  $s$ -oblivious RBC of sequences (with sampling, after preprocessing)

---

**Input:**  $G(V, E), R, T, \rho, S = \{s_0, \dots, s_k\}, \delta_{s,t}(v), \delta_{\bullet,t}(v)$

**Output:**  $RBC$

**Data:**  $\delta$

$RBC = 0$

**for**  $t \in V$  **do**

$\delta = \delta_{\bullet,t}(s_0) \cdot \rho_{s_0}$   
    **for**  $i = 0$  **to**  $k - 1$  **do**  
         $\delta = \delta_{s_i,t}(s_{i+1}) \cdot \rho_{s_{i+1}};$   
     $RBC += \delta;$

**return**  $RBC$

---

**Algorithm 7.**  $s$ -oblivious RBC of sets (with sampling)

---

**Input:**  $G(V, E), R, T, \rho, M$   
**Output:**  $RBC$   
**Data:**  $delta[1..|V|]$   
 $RBC = 0$   
**for**  $t \in V$  **do**  
    **► topological sort**  
    **▼ init delta**  
    |  $\forall v \in V, delta[v] = (1 - \rho_v) \cdot T(v, t);$   
    **► accumulate**  $\delta_{\bullet, \bullet}(M_\rho);$   
**return**  $RBC$

---

The following equations describe RBC of the fuzzy set  $M_\rho$  in terms of  $\lambda_{\bullet, t}^{M_\rho}(v)$ :

$$\lambda_{\bullet, t}^{M_\rho}(v) = (1 - \rho_v) \cdot T(v, t) + \sum_{u \in Pred_t(v)} \lambda_{\bullet, t}^{M_\rho}(u) \cdot R(\odot, u, v, t) \cdot (1 - \rho_v) \quad (18)$$

$$\delta_{\bullet, \bullet}(M_\rho) = \sum_{t \in V} \left( \sum_{s \in V} T(s, t) - \lambda_{\bullet, t}^{M_\rho}(t) \right). \quad (19)$$

Algorithm 7 computes RBC of a set of monitors installed on vertices in a communication network with source-oblivious routing strategy, given the sampling rates of the monitors. Thus, the time complexity of Algorithm is  $O(nm)$ . This algorithm iterates over all vertices in the network and in each iteration, sorts vertices in the same way as Algorithm 4. It initializes each entry of the  $delta[v]$  array to  $(1 - \rho_v) \cdot T(v, t)$  and accumulates betweenness similarly to Algorithm 3

*Contribution to RBC of a Set.* In this section, we assume that a set of monitors  $X$  is installed on vertices in a network and their sampling rates are specified by  $\rho$ . We investigate the expected number of unsampled packets that can be sampled by additional monitors. We will refer to this measure as the contribution of individual vertices, sets of vertices, or sequences of vertices to RBC of  $X_\rho$ . In Section 4.3, we have defined  $\lambda_{s, t}^{X_\rho}(v)$  as the probability that a packet from  $s$  to  $t$  will pass through  $v$  without being sampled by monitors in  $X_\rho$ . This probability gives no information regarding the probability that this packet will be sampled after passing through  $v$ .

Let  $\chi_{s, t}^{X_\rho}(w)$  be the probability that a packet from  $s$  to  $t$  will pass through  $w$  and will not be sampled by any of the monitors in  $X_\rho$  (neither before nor after visiting  $w$ ). Assume that  $v$  monitors the traffic with sampling rate  $\rho_w > 0$ . Then  $\chi_{s, t}^{X_\rho}(w) \cdot \rho_v \cdot T(s, t)$  is the expected number of packets from  $s$  to  $t$  that were sampled only by  $w$  and not by other monitors. In other words,  $\chi_{s, t}^{X_\rho}(w) \cdot \rho_w$  is the contribution of  $w$  to the capability of the monitors to sample traffic between  $s$  and  $t$ .  $\chi_{s, t}^{X_\rho}(u)$  can be computed for any  $u \in V$  by starting with  $X = \emptyset$  and adding vertices to  $X$  one at a time using the following lemma:

**LEMMA 2 (PAIRWISE DEPENDENCY CONTRIBUTION).** *Let  $X = \{v_1, \dots, v_k\}$  be a set of vertices with sampling rates specified by  $\rho_{v_1}, \dots, \rho_{v_k}$ , respectively. Let  $w$*

be a vertex with sampling rate  $\rho_w$ . For any  $u \in V$ , it holds that:

$$\begin{aligned} (u = w) \chi_{s,t}^{X_\rho \cup \{w\}_\rho}(u) &= \chi_{s,t}^{X_\rho}(u) \cdot (1 - \rho_w) \\ (u \neq w) \chi_{s,t}^{X_\rho \cup \{w\}_\rho}(u) &= \chi_{s,t}^{X_\rho}(u) - \chi_{s,t}^{X_\rho}(u) \cdot \chi_{u,t}^{X_\rho}(w) \cdot \rho_w - \chi_{s,t}^{X_\rho}(w) \cdot \chi_{w,t}^{X_\rho}(u) \cdot \rho_w \end{aligned} \quad (20)$$

PROOF. This lemma describes the computation of the probability that a packet from  $s$  to  $t$  will pass through  $u$  without being sampled neither by monitors in  $X$  nor by  $w$ . The guiding principle of the computation is: to discard packets that were sampled by  $w$  we need to subtract from  $\chi_{s,t}^{X_\rho}(u)$  the probability of the packet being sampled by  $w$  either before or after passing through  $u$ .

The probability of a packet passing through  $w$  without being sampled by  $w$  or any other vertex in  $X$  ( $\tilde{\chi}_{s,t}^{X \cup \{w\}}(w)$ ), equals its probability of passing through  $w$  without being sampled by any vertex in  $X$  and not being sampled by  $w$ . Being sampled by  $w$  and being sampled by any vertex in  $X$  are independent events (assuming that  $w \notin X$ ). Hence, the first case of the lemma.

Assume  $w \neq u$ . Let the event  $Y_v$  represent the cases where the packet from  $s$  to  $t$  passes through the vertex  $v$ . Let the event  $Z_v$  represent the cases where this packet was sampled by the vertex  $v$ . Let  $\bar{Z}_v$  represent the cases where this packet was not sampled by the vertex  $v$ .  $\chi_{s,t}^{X_\rho}(u)$  is the probability that a packet from  $s$  to  $t$  was not sampled by any vertex in  $X$ , but passes through  $u$ :

$$\chi_{s,t}^{X_\rho}(u) = Pr \left[ \bigcap_{v \in X} \bar{Z}_v \cap Y_u \right]. \quad (21)$$

$Pr[\bigcap_{v \in X} \bar{Z}_v \cap Y_u]$  can be decomposed into two cases: packets that were sampled by  $w$  and packets that were not:

$$\chi_{s,t}^{X_\rho}(u) = Pr \left[ \bigcap_{v \in X} \bar{Z}_v \cap Z_w \cap Y_u \right] + Pr \left[ \bigcap_{v \in X} \bar{Z}_v \cap \bar{Z}_w \cap Y_u \right]. \quad (22)$$

Assume a packet from  $s$  to  $t$  that was not sampled by any vertex in  $X$ . The above equation yields that the probability of this packet passing through  $u$  without being sampled by  $w$  (case  $\bar{Z}_w \cap Y_u$ ) is equal to the probability of the packet passing through  $u$  minus the probability of the packet passing through  $u$  while being sampled by  $w$  (case  $Z_w \cap Y_u$ ).

In accordance with Proposition 1 the packet can pass through  $u$  and  $w$  by either passing first through  $u$  and then through  $w$ , or vice-versa. Note that packet can be sampled by  $w$  only if it passes through  $w$ . Therefore, the case  $Z_w \cap Y_u$  can be represented as the sum of two sequence dependencies multiplied by the sampling rate of  $w$  ( $\tilde{\delta}_{s,t}((w, u)) \cdot \rho_w + \tilde{\delta}_{s,t}((u, w)) \cdot \rho_w$ ). Moreover, the proof of Proposition 1 can easily be translated from  $\delta$  to  $\chi$  by excluding packets that are sampled by some vertex in  $X$ .

By replacing the probabilities in Eq. (22) with the respective pairwise dependencies, we get:

$$\chi_{s,t}^{X_\rho}(u) = (\tilde{\chi}_{s,t}^{X_\rho}((u, w)) \cdot \rho_w + \tilde{\chi}_{s,t}^{X_\rho}((w, u)) \cdot \rho_w) + \chi_{s,t}^{X_\rho \cup \{w\}_\rho}(u). \quad (23)$$

In accordance with to the Dependency Chaining Lemma, which can also be adjusted to  $\chi$  by considering only packets that were not sampled by  $X$ ,  $\tilde{\chi}_{s,t}^{X_\rho}((u, w))$  can be decomposed into a product of two pairwise dependencies, completing the proof.

$$\chi_{s,t}^X(u) - \chi_{s,t}^X(u) \cdot \chi_{u,t}^X(w) \cdot \rho_w - \chi_{s,t}^X(w) \cdot \chi_{w,t}^X(u) \cdot \rho_w = \chi_{s,t}^{X \cup \{w\}}(w). \quad \square \quad (24)$$

Let  $\chi_{\bullet,t}^{X_\rho}(u) = \sum_{s \in V} \chi_{s,t}^{X_\rho}(u) \cdot T(s, t)$  be the expected number of packets targeted at  $t$  that will pass through  $v$  and will not be sampled by any of the monitors in  $X_\rho$ . Lemma 2 can be used to compute  $\chi_{\bullet,t}^{X_\rho \cup \{w\}}(u)$  by multiplying Eq. (20) by  $T(s, t)$  and summing it over all sources.

$$\begin{aligned} (u = w) \quad \chi_{\bullet,t}^{X_\rho \cup \{w\}}(u) &= \chi_{\bullet,t}^{X_\rho}(u)(1 - \rho_w) \\ (u \neq w) \quad \chi_{\bullet,t}^{X_\rho \cup \{w\}}(u) &= \chi_{\bullet,t}^{X_\rho}(u) - \chi_{\bullet,t}^{X_\rho}(u) \cdot \chi_{u,t}^{X_\rho}(w) \cdot \rho_w - \chi_{\bullet,t}^{X_\rho}(w) \cdot \chi_{w,t}^{X_\rho}(u) \cdot \rho_w \end{aligned} \quad (25)$$

Let  $\chi_{\bullet,\bullet}^{X_\rho}(w) = \sum_{t \in V} \chi_{\bullet,t}^{X_\rho}(w)$  be the expected number of packets between all source-target pairs that pass through  $w$  without being sampled by any vertex in  $X$ .  $\chi_{\bullet,\bullet}^{X_\rho}(w) \cdot \rho_w$  can be considered as the contribution of  $w$  to RBC of  $X_\rho$ :

$$\ddot{\delta}_{\bullet,\bullet}(X_\rho \cup \{w\}_\rho) = \ddot{\delta}_{\bullet,\bullet}(X_\rho) + \chi_{\bullet,\bullet}^{X_\rho}(w) \cdot \rho_w.$$

*Using Precomputed Data.* We assume that all  $\delta_{s,t}(v)$ ,  $\delta_{\bullet,t}(v)$ , and  $\delta_{\bullet,\bullet}(v)$  values are computed using Algorithm 1 and stored in a data structure with  $O(1)$  store and retrieval, such as matrices or a hash table. The computation speed up methods presented here are valid for source-oblivious routing strategies. In particular, we assume that all routing decisions specified by the probabilities  $R(\emptyset, u, v, t)$  are independent.

In order to make the discussion more intuitive, we will use an example from set theory. Let  $A$ ,  $B$ , and  $C$  be three sets. We need to compute the size of their union, namely, the number of elements belonging to at least one of the sets. Assume that we can easily compute the size of intersection but not the size of union. In order to overcome this difficulty, we can use the Inclusion-Exclusion rule as following:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|.$$

Regrouping the addends will result in:

$$|A \cup B \cup C| = |A| + |B \cap \overline{A}| + |C \cap \overline{A} \cap \overline{B}|,$$

where  $|B \cap \overline{A}| = |B| - |A \cap B|$  accounts for all elements that belong to  $B$  and do not belong to  $A$ .

In our case the size of a union can be associated with RBC of sets, where we account for packets sampled by at least one monitor in the set. Size of an intersection can be associated with RBC of sequences, where we account for packets sampled by all monitors in the sequence. Finally, the semantics of  $|B \cap \overline{A}|$  are similar to semantics of  $\chi_{s,t}^{\{u\}}(v)$  – the probability of a packet from  $s$  to  $t$  to pass through  $v$  without passing through  $u$ . Next we will apply the technique demonstrated in the above example for computing the expected number of packets sampled by a set of monitors. Pairwise dependency on a set of monitors can be computed by summing contributions of the set members as described by the following lemma:

LEMMA 3 (SUMMING DEPENDENCY CONTRIBUTIONS). *Let  $M = \{v_0, \dots, v_k\}$  be a set of vertices with sampling rates specified by  $\rho_{v_0}, \dots, \rho_{v_k}$ , respectively. Let  $M^{(i)} = \{v_0, \dots, v_i\}$  be a subset of  $M$ . It holds that:*

$$\ddot{\delta}_{s,t}(M_\rho) = \sum_{i=0}^k \chi_{s,t}^{M_\rho^{(i-1)}}(v_i) \cdot \rho_{v_i}.$$

PROOF. Lemma 3 describes an iterative computation  $\ddot{\delta}_{s,t}(M_\rho)$ . In each iteration we accumulate the contributions of  $v_i$ , namely the uncovered traffic flows sampled by  $v_i$ , to the pairwise dependency.

Let the event  $Z_v$  represent all cases where the packet from  $s$  to  $t$  is sampled by vertex  $v$ . Let the event  $\overline{Z}_v$  represent all cases where the packet from  $s$  to  $t$  is not sampled by the vertex  $v$ . Let the event  $Y_v$  represent all cases where the packet from  $s$  to  $t$  passes through vertex  $v$ . By definition,  $\ddot{\delta}_{s,t}(\{v_1, \dots, v_k\})$  is the probability that the packet, from  $s$  to  $t$ , will pass through at least one of the  $k$  vertices  $v_1, \dots, v_k$

$$\ddot{\delta}_{s,t}(\{v_1, \dots, v_k\}_\rho) = Pr \left[ \bigcup_{i=1}^k Z_{v_i} \right].$$

We can substitute the right term in the above equation by a summation as following:

$$\ddot{\delta}_{s,t}(\{v_1, \dots, v_k\}_\rho) = \sum_{i=1}^k Pr \left[ Z_{v_i} \cap \bigcap_{j=1}^{i-1} \overline{Z}_{v_j} \right].$$

For each  $i$  the term inside the summation above is the probability that the packet from  $s$  to  $t$  will be sampled by  $v_i$  without being sampled by  $v_1, \dots, v_{i-1}$ . In accordance with the definition of  $\chi$

$$Pr \left[ Z_{v_i} \cap \bigcap_{j=1}^{i-1} \overline{Z}_{v_j} \right] = \chi^{\{v_1, \dots, v_{i-1}\}}(v_i)$$

The sampling rate of the vertex  $v_i$  is  $\rho_{v_i}$  and therefore we can substitute the term inside the summation by  $\chi^{\{v_1, \dots, v_{i-1}\}}(v_i) \cdot \rho_{v_i}$  completing the proof:

$$\ddot{\delta}_{s,t}(\{v_1, \dots, v_k\}) = \sum_{i=1}^k \delta^{\{v_1, \dots, v_{i-1}\}}(v_i) \cdot \rho_{v_i}. \quad \square$$

Lemma 3 provides us with a tool for iterative computation of pairwise dependency on sets of vertices. Summing  $\ddot{\delta}_{s,t}(M_\rho)$  over all sources while multiplying each addend by  $T(s, t)$  results in Eq. (26) that describes iterative computation of target dependency on a set of vertices.

$$\ddot{\delta}_{\bullet,t}(M_\rho) = \sum_{i=0}^k \chi_{\bullet,t}^{M_\rho^{(i-1)}}(v_i) \cdot \rho_{v_i}. \quad (26)$$

**Algorithm 8.**  $s$ -oblivious RBC of sets (with sampling, after preprocessing)**Input:**  $G(V, E)$ ,  $R$ ,  $T$ ,  $\rho$ ,  $M = \{v_0, \dots, v_k\}$ ,  $\delta_{s,t}(v)$ ,  $\delta_{\bullet,t}(v)$ **Output:**  $RBC$ **Data:**  $pdep[k \times k \times n]$ ,  $tdep[k \times n]$ ,  
 $npdep[k \times k \times n]$ ,  $ntdep[k \times n]$ , $RBC = 0$ **for**  $s, v \in M$ ,  $t \in V$  **do**  $pdep[s, v, t] = \delta_{s,t}(v)$ **for**  $v \in M$ ,  $t \in V$  **do**  $tdep[v, t] = \delta_{\bullet,t}(v)$ **▼account for  $M$** **for**  $w \in M$  **do****for**  $t \in V$  **do** $RBC += tdep[w, t] \cdot \rho_w$ **for**  $u \in M$  **do****if**  $u = w$  **then** $ntdep[u, t] = tdep[u, t] \cdot (1 - \rho_w)$ **else** $ntdep[u, t] = tdep[u, t] -$  $-tdep[u, t] \cdot pdep[u, w, t] \cdot \rho_w -$  $-tdep[w, t] \cdot pdep[w, u, t] \cdot \rho_w;$ **for**  $s \in M$  **do****if**  $u = w$  **then** $npdep[s, u, t] = pdep[s, u, t] \cdot (1 - \rho_w)$ **else** $npdep[s, u, t] = pdep[s, u, t] -$  $-pdep[s, u, t] \cdot pdep[u, w, t] \cdot \rho_w -$  $-pdep[s, w, t] \cdot pdep[w, u, t] \cdot \rho_w;$  $tdep = ntdep$ ;  $pdep = npdep$ ;**return**  $RBC$ 

Summing Eq. (26) over all targets results in iterative computation of RBC of a set of vertices:

$$\ddot{\delta}_{\bullet,\bullet}(M_\rho) = \sum_{t \in V} \sum_{i=0}^k \chi_{\bullet,t}^{M_\rho^{(i-1)}}(v_i) \cdot \rho_{v_i}. \quad (27)$$

In the last algorithm presented in this article (Algorithm 8), we compute RBC of a given set iterating over all vertices in the set and summing their marginal contributions as described in Eq. (27). The marginal contributions are computed using Lemma 2. During the algorithm we maintain two matrices. One is the three-dimensional matrix of pairwise dependencies and the other is the two-dimensional matrix of target dependencies. The last dimension in these matrices is of size  $n$  while the other dimensions are of size  $k$ . We use Eq. (20) to update pairwise dependencies and Eq. (25) to update target dependencies.

Algorithm 8 is composed of an initialization phase where precomputed values are copied into temporal matrices and four nested loops that compute RBC of the input set of vertices. Temporal arrays  $pdep$  and  $tdep$  maintain pairwise and target dependencies respectively. Initially the values in these arrays correspond to  $\chi_{s,t}^{\emptyset}(v) = \delta_{s,t}(v)$  and  $\chi_{\bullet,t}^{\emptyset}(v) = \delta_{\bullet,t}(v)$ . In each iteration of the outer loop, we process one vertex from the input set  $M$ . The marginal contributions of vertices to RBC of  $M$  are aggregated according to Eq. (27) during the first inner loop (that

iterates over all  $t \in V$ ). The second and third inner loops iterate over vertices in  $M$  and update the entries of the  $tdep$  and  $pdep$  matrices, respectively, in accordance with Eqs. (25) and (20). After the first update of all the values in these matrices, they correspond to  $\chi_{s,t}^{M^{(1)}}(v)$  and  $\chi_{\bullet,t}^{M^{(1)}}(v)$  where  $M^{(1)}$  contains only the first vertex in  $M$ . During the second iteration, we process one more vertex from  $M$  and update the matrices again and so on until we process all vertices in  $M$ . The overall time complexity of Algorithm 6 is  $O(k^3n)$  where  $k$  is the size of  $M$  and  $n$  is the number of vertices in the network.

**5.4. RBC OF SETS OF EDGES AND MIXED SETS.** In many applications, the monitoring of the traffic is done by tapping the communication links and not the vertices. The problem of monitoring links can easily be reduced to a problem of monitoring vertices by adding a phantom vertex in the middle of the monitored link. When the routing scheme  $R$  and the traffic matrix  $T$  are updated appropriately.

Algorithms that do not use pre-processing can be configured to avoid the phantom vertices in their main loop. This is an intuitive optimization since they do not introduce traffic to the network. In this case, adding phantom vertices does not increase the complexity of these algorithms but makes each iteration of the main loop twice as long as before. In algorithms that use pre-processing, after adding phantom vertices, the size of the pre-computed matrices will depend on the number of edges and not only the number of vertices. Thus, adding phantom vertices increases both time and space complexity of the preprocessing stage in Algorithms 6 and 6 from  $O(n^2m)$  time and  $O(n^3)$  space to  $O(|E|^2m)$  time and  $O(|E|^3)$  space.

In order to avoid the addition of phantom vertices, we should remember that the RBC of a (directed) link is the RBC of the sequence consisting of both of its' vertices. Therefore, the RBC of a sequence of directed links is RBC of the sequence of vertices comprising the links and can be computed using Algorithms 2, 5, and 6. The RBC of a set of links can be computed iteratively, by taking into account one link at a time, similarly to RBC of a set of vertices.

Let  $(u, v)$  be a link tapped by a monitor with sampling rate  $\rho_{(u,v)}$ . The expected number of packets that will pass through  $w$  and will not be sampled by that monitor is equal to  $\delta_{\bullet,\bullet}(w)$  minus RBC of sequences  $(w, u, v)$  and  $(u, v, w)$ . Note that if the sequence  $(u, v)$  represents a link and  $u \neq w \neq v$ , then the RBC of the sequence  $(u, w, v)$  is zero. We can modify the recursive Eq. (20) to compute the probability that a packet from  $s$  to  $t$  will pass through  $w$  and will not be sampled by any link monitor in a given set. Let  $Q^{(i)} = \{(u_1, v_1), \dots, (u_i, v_i)\}$  be a set of vertices with sampling rates specified by  $\rho_{(v_1, u_1)}, \dots, \rho_{(u_i, v_i)}$ , respectively.

$$\chi_{s,t}^{Q^{(i+1)}}(w) = \chi_{s,t}^{Q^{(i)}}(w) - \tilde{\chi}_{s,t}^{Q^{(i)}}((w, u, v)) - \tilde{\chi}_{s,t}^{Q^{(i)}}((u, v, w)) \quad (28)$$

where  $\tilde{\chi}_{s,t}^{Q^{(i)}}((w, u, v)) = \chi_{s,t}^{Q^{(i)}}(w) \cdot \chi_{w,t}^{Q^{(i)}}(u_{i+1}) \cdot \chi_{u_{i+1},t}^{Q^{(i)}}(v_{i+1})$  and  $\tilde{\chi}_{s,t}^{Q^{(i)}}((u, v, w)) = \chi_{s,t}^{Q^{(i)}}(u_{i+1}) \cdot \chi_{u_{i+1},t}^{Q^{(i)}}(v_{i+1}) \cdot \chi_{v_{i+1},t}^{Q^{(i)}}(w)$ . Target dependency with respect to a set of link monitors ( $\chi_{\bullet,\bullet}^{Q^{(i+1)}}(w)$ ) can be computed using the equation above if we substitute pairwise dependencies with target dependencies.

Algorithm 9 computes the RBC of a set of links similarly to the way that Algorithm 8 computes the RBC of a set of vertices. But instead of implementing

**Algorithm 9.**  $s$ -oblivious RBC of sets of links (with sampling, after preprocessing)**Input:**  $G(V, E), R, T, \rho, Q = \{(u_1, v_1), \dots, (u_k, v_k)\}, \delta_{s,t}(v), \delta_{\bullet,t}(v)$ **Output:**  $RBC$ **Data:**  $pdep, npdep, tdep, ntdep$  $RBC = 0$  $X =$  set of vertices comprising  $Q$ **for**  $s, v \in X, t \in V$  **do**  $pdep[s, v, t] = \delta_{s,t}(v)$ **for**  $v \in X, t \in V$  **do**  $tdep[v, t] = \delta_{\bullet,t}(v)$ **▼ account for  $Q$** **for**  $(u, v) \in Q$  **do****for**  $t \in V$  **do** $RBC+ = tdep[u, t] \cdot pdep[u, v, t]$ **for**  $w \in X$  **do** $ntdep[w, t] = tdep[w, t] -$  $- tdep[w, t] \cdot pdep[w, u, t] \cdot pdep[u, v, t] -$  $- tdep[w, t] \cdot pdep[w, u, t] \cdot pdep[u, v, t]$ **for**  $s \in X$  **do** $npdep[s, w, t] = pdep[s, w, t] -$  $- pdep[s, w, t] \cdot pdep[w, u, t] \cdot pdep[u, v, t] -$  $- pdep[s, w, t] \cdot pdep[w, u, t] \cdot pdep[u, v, t];$  $tdep = ntdep; pdep = npdep;$ **return**  $RBC$ **Algorithm 10.**  $s$ -oblivious RBC of sets of links and vertices (with sampling, after preprocessing)**Input:**  $G(V, E), R, T, \rho, M, Q, \delta_{s,t}(v),$  $\delta_{\bullet,t}(v)$ **Output:**  $RBC$ **Data:**  $pdep, npdep, tdep, ntdep$  $RBC = 0$  $X =$  set of vertices comprising  $Q$  and  $M$ **for**  $s, v \in X, t \in V$  **do**  $pdep[s, v, t] = \delta_{s,t}(v)$ **for**  $v \in X, t \in V$  **do**  $tdep[v, t] = \delta_{\bullet,t}(v)$ **► account for  $M$** **► account for  $Q$** **return**  $RBC$ 

the Eqs. (20) and (25), it implements the Eq. (28) and its target dependency variant. Another difference between the Algorithms 9 and 8 is the size of the  $pdep$  and  $tdep$  matrices. The size of the first dimensions of these matrices is now equal to the number of vertices attached to the input links.

Algorithm 9 has the same four nested loops as Algorithm 8. In addition the number of vertices comprising the links in a given set of size  $k$  is at least  $k + 1$  and at most  $2k$ . Therefore the time complexity and the space complexity of Algorithm 9 is  $O(k^3n)$  and  $O(k^2n)$  respectively as well as to Algorithm 8.

Both algorithms can be combined together in order to compute the RBC of a set of monitors that includes monitors installed on vertices and monitors installed on links as shown in Algorithm 10. Let  $M$  be the set of monitored vertices and  $Q$  be the set of monitored links. Let  $X = M \cup \{u, v : (u, v) \in Q\}$  be the set of vertices comprising  $M$  and  $Q$ . First all the data relevant to  $X$  is copied into the  $tdep$  and  $pdep$  matrices. Afterwards, the cores of both algorithms are executed consequently to compute RBC of the mixed set  $M \cup Q$ .



## 6. Conclusions

In this article, we have defined a new Betweenness-Centrality measure called Routing Betweenness-Centrality (RBC) which is a generalization of well known betweenness centrality measures such as Shortest-Path Betweenness Centrality, Traffic Load Centrality, and Flow Betweenness Centrality. RBC measures the extent to which vertices or groups of vertices are exposed to the traffic given any loop-free routing strategy.

The algorithms presented in this article are easily modified to compute RBC of groups consisting of links and/or vertices (see Appendix 5.4). In fact, a more sophisticated combinations of policies for traffic monitoring/controlling are supported. Using the methods present in this article we can compute the expected number of packets each one of which satisfies a predicate in disjunctive normal form with at most one negation clause. For example, packets each one of which is sampled by  $q$ ,  $u$ , and  $v$ , or by  $w$  and  $x$  but is neither sampled by  $y$  nor by  $w$ .

The required computation complexity of our algorithms depend on whether the routing scheme is source dependent or source oblivious. Generally speaking, when the routing decisions in the network depend on both the source and the target of a packet the time complexity of RBC computation is an order of  $n$  higher than in the source-oblivious cases. For source oblivious routing schemes, our RBC algorithms can be used to compute the Shortest Path Betweenness-Centrality and Traffic Load Centrality with complexity matching the state of the art complexities; while our RBC algorithms are capable to compute a larger variety of Betweenness-Centrality measures.

We show that preprocessing can dramatically reduce the time required for a single computation of RBC of a sequence. Preprocessing can also reduce the time required to compute the RBC of sets, when the size of the investigated set is smaller than the third square of  $m$  (the number of edges in the routing tree or the routing DAG of the given routing scheme).

In the domain of communication networks, RBC is useful for predicting the effectiveness and the cost of passive network monitoring. RBC can be used in conjunction with various combinatorial optimization techniques and approximation algorithms [Suh et al. 2006; Puzis et al. 2007b; Dolev et al. 2009], for optimizing the placement of passive monitors within the communication network. Other applications include simulation free prediction of congestions in communication networks, design and examination of routing strategies, and optimizing network layout for, as an example, balancing the traffic load in a network and assuring service level agreements.

Although RBC is described in terms of communication networks, it can easily be applied for a wide variety of network related tasks. For example, RBC is applicable to Markov chains and probabilistic automata, assuming that their execution is loop-less (i.e., you can not visit the same vertex twice during a single execution). Here, RBC can be used for computing the probability of visiting a group of states during an execution. Specific variants of SPBC such as  $k$ -betweenness and  $l$ -length scaled betweenness described by Borgatti and Everett [2006] can be implemented using RBC. In the later case each entry in the traffic matrix should be inversely proportional to the distance between source and target. The routing function can be specially designed to allow amplification or reduction of the traffic as it flows from the source to the target. Along with a specially designed traffic matrix, this

allows the use of RBC algorithms for computing the linearly scaled variant of SPBC [Geisberger et al. 2008; Brandes 2008].

The comprehensiveness of RBC is yet to be fully understood at this point. Future research can focus on both the exploitation of RBC in network analysis, and the further generalization of RBC to include random walks and other transport mechanisms. We believe that the presented concept of RBC, and algorithms for its computation will help researchers and practitioners in the analysis of complex networks in general and communication networks in particular.

**ACKNOWLEDGMENT.** The authors would like to thank Omer Zohar for implementing and testing all RBC algorithms and department members who contributed valuable remarks on this paper.

## REFERENCES

- ANTHONISSE, J. M. 1971. The rush in a directed graph. Tech. rep. BN 9/71, Stichting Mathematisch Centrum, Amsterdam, The Netherlands.
- BARABASI, A.-L., AND ALBERT, R. 1999. Emergence of scaling in random networks. *Science* 286, 509–512.
- BARABASI, A.-L., ALBERT, R., AND JEONG, H. 2000. Scale-free characteristics of random networks: the topology of the world-wide web. *Phys. A* 281, 69–77.
- BARTHÉLEMY, M. 2004. Betweenness centrality in large complex networks. *The Europ. Phys. J. B – Condensed Matter* 38, 2 (Mar.), 163–168.
- BOLLOBAS, B., AND RIORDAN, O. 2003. Robustness and vulnerability of scale-free random graphs. *Internet Math.* 1, 1, 1–35.
- BORGATTI, S. P. 2005. Centrality and network flow. *Social Netw.* 27, 55–71.
- BORGATTI, S. P., AND EVERETT, M. G. 2006. A graph-theoretic perspective on centrality. *Social Netw.* 28, 4 (Oct.), 466–484.
- BORK, P., JENSEN, L. J., VON MERING, C., RAMANI, A. K., LEE, I., AND MARCOTTE, E. M. 2004. Protein interaction networks from yeast to human. *Curr. Opin. Struct. Biol.* 14, 3 (Jun.), 292–299.
- BRANDES, U. 2001. A faster algorithm for betweenness centrality. *Math. Soc.* 25, 2, 163–177.
- BRANDES, U. 2008. On variants of shortest-path betweenness centrality and their generic computation. *Social Netw.* 30, 2, 136–145.
- CANTIENI, G. R., IANNACONE, G., BARAKAT, C., DIOT, C., AND THIRAN, P. 2006. Reformulating the monitor placement problem: optimal network-wide sampling. In *Proceedings of the ACM CoNEXT Conference (CoNEXT'06)*. ACM, New York, 1–12.
- DOLEV, S., ELOVICI, Y., PUZIS, R., AND ZILBERMAN, P. 2009. Incremental deployment of network monitors based on group betweenness centrality. *Inf. Proc. Lett.* 109, 20, 1172–1176.
- EVERETT, M. G., AND BORGATTI, S. P. 1999. The centrality of groups and classes. *Math. Soc.* 23, 3, 181–201.
- FALOUTSOS, M., FALOUTSOS, P., AND FALOUTSOS, C. 1999. On power-law relationships of the internet topology. *SIGCOMM Comput. Comm. Rev.* 29, 4, 251–262.
- FREEMAN, L. C. 1977. A set of measures of centrality based on betweenness. *Sociometry* 40, 1, 35–41.
- FREEMAN, L. C. 1979. Centrality in social networks conceptual clarification. *Social Netw.* 1, 215–239.
- FREEMAN, L. C., BORGATTI, S. P., AND WHITE, D. R. 1991. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Netw.* 13, 2 (Jun.), 141–154.
- GEISBERGER, R., SANDERS, P., AND SCHULTES, D. 2008. Better approximation of betweenness centrality. In *Proceedings of the 8th Workshop on Algorithm Engineering and Experimentation (ALENEX08)*. SIAM, Philadelphia, PA.
- GOH, K.-I., KAHNG, B., AND KIM, D. 2001. Universal behavior of load distribution in scale-free networks. *Phys. Rev. Lett.* 87, 27 (Dec.), 278701.
- HARARY, F., NORMAN, R., AND CARTWRIGHT, D. 1965. *Structural Models. An Introduction to the Theory of Directed Graphs*. Wiley, New York.
- HOLME, P. 2003. Congestion and centrality in traffic flow on complex networks. *Adv. Complex Syst.* 6, 2, 163–176.
- JACKSON, A., MILLIKEN, W., SANTIVANEZ, C., CONDELL, M., AND STRAYER, W. 2007. A topological analysis of monitor placement. In *Proceedings of the 6th IEEE International Symposium on Network*

- Computing and Applications (NCA 2007)*. IEEE Computer Society Press, Los Alamitos, CA, 169–178.
- MOY, J. 1998. Rfc 2328 - ospf version 2. <http://www.ietf.org/rfc/rfc2328.txt>.
- NEWMAN, M. E. J. 2001. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E* 64, 016132.
- NEWMAN, M. E. J. 2005. A measure of betweenness centrality based on random walks. *Soc. Netw.* 27, 1 (Jan.), 39–54.
- PASTOR-SATORRAS, R., AND VESPIGNANI, A. 2002. Immunization of complex networks. *Phys. Rev. E* 65, 036104.
- PORTA, S., CRUCITTI, P., AND LATORA, V. 2006. The network analysis of urban streets: a primal approach. *Environment and Plan. B: Planning and Design* 33 5 (Sept.), 705–725.
- PUZIS, R., ELOVICI, Y., AND DOLEV, S. 2007a. Fast algorithm for successive computation of group betweenness centrality. *Phys. Rev. E* 76, 5, 056709.
- PUZIS, R., ELOVICI, Y., AND DOLEV, S. 2007b. Finding the most prominent group in complex networks. *AI Comm.* 20, 287–296.
- PUZIS, R., KLIPPEL, M. D., ELOVICI, Y., AND DOLEV, S. 2007c. Optimization of NIDS placement for protection of intercommunicating critical infrastructures. In *EuroISI*. (Esbjerg, Denmark) Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, (Esbjerg, Denmark).
- PUZIS, R., YAGIL, D., ELOVICI, Y., AND BRAHA, D. 2009. Collaborative attack on internet users' anonymity. *Internet Res.* 19, 1, 60–77.
- STROGATZ, S. H. 2001. Exploring complex networks. *Nature* 410, 268–276.
- SUH, K., GUO, Y., KUROSE, J., AND TOWSLEY, D. 2006. Locating network monitors: Complexity, heuristics, and coverage. *Comput. Commun.* 29, 1564–1577.
- VILLAMIZAR, C. 2002. OSPF optimized multipath (OSPF-OMP). <http://www.faster-light.net/ospf-omp/ospf-omp.pdf>.
- WASSERMAN, S., AND FAUST, K. 1994. *Social network analysis: Methods and applications*. Cambridge University Press., Cambridge, England.
- YAGIL, D. 2005. Collaborative attack on www users' anonymity. M.S. dissertation, Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
- YAN, G., ZHOU, T., HU, B., FU, Z.-Q., AND WANG, B.-H. 2006. Efficient routing on complex networks. *Phys. Rev. E* 73, 046108.
- YOOK, S.-H., JEONG, H., AND BARABASI, A.-L. 2002. Modeling the internet's large-scale topology. *Proc. Nat. Acad. Sci.* 99, 21 (Oct.), 13382–13386.
- ZHOU, T., LIU, J.-G., AND WANG, B.-H. 2006. Notes on the algorithm for calculating betweenness. *Chinese Phys. Lett.* 23, 2327–2329.

RECEIVED JULY 2009; REVISED FEBRUARY 2010; ACCEPTED FEBRUARY 2010