Routing Betweenness Centrality

by

Shlomi Dolev, Yuval Elovici, and Rami Puzis

Technical Report #2009-09

August 2009

Routing Betweenness Centrality

Shlomi Dolev, Yuval Elovici, and Rami Puzis
Ben-Gurion University of the Negev

August 10, 2009

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 paper we define the Routing Betweenness Centrality (RBC) measure which 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 sequences 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 in the number of vertices in the sequence or the set.

1 Introduction

Networks are commonly used to represent a domain, a problem, or a complex dynamic system in a large variety of scopes [31]. Representing, for example, social networks [33], protein interactions [9], urban structure [27], and computer communication networks [15, 35]. Various centrality measures such as Degree, Closeness, and Betweenness [8, 17] were introduced in order to analyze networks and understand both the global dynamics of the networks and the roles played by individual nodes. Many naturally evolved complex networks are characterized by a power-law distribution of Degree and Betweenness-Centrality measures of their nodes [5, 15]. Such networks, referred to as Scale-Free Networks [3, 4], are highly resistant to random damages but are easily partitioned by removing the most central nodes [6].

The centrality characteristics of nodes are also important in the field of computational epidemiology, where it has been shown that immunizing central nodes can significantly reduce the impact of epidemics [26]. In the scope of the Internet, Jackson et al. [22] suggest placing monitors on links of the autonomous system level topology of the Internet, with end nodes having the highest Degree. The computation of Group Betweenness-Centrality was suggested in [30] to identify groups of autonomous systems which can collaborate to trace the communications of as many Internet users as possible.

In this paper we concentrate on Betweenness-Centrality measures [2, 16], originally defined 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 nodes in communication networks. The concept of Betweenness-Centrality evolved into a broad class of diverse measures that consider different types of network flows [7]. Betweenness, which is now referred to as Shortest Path Betweenness-Centrality (SPBC) assumes that only shortest paths are used to transfer the network flow. Traffic Load Centrality (TLC) [11, 19, 24] 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 TLC.

Table 1: Time and space complexity of the proposed algorithms.

Section / Alg.	**Scope**	**Routing depends on**	**Space**	**Preproc. time**	**Query time**
4.1 / 1	All nodes	source and target	$O(m)$		$O(n^2m)$
4.2 / 2	Sequence	source and target	$O(m)$		$O(n^2m)$
4.3 / 3	Set (nodes)	source and target	$O(m)$		$O(n^2m)$
5.1 / 4	All nodes	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 (nodes)	only target	$O(m)$		$O(nm)$
5.3 / 8	Set (nodes)	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 nodes 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 nodes in a sequence or a set.

Flow Betweenness-Centrality (FBC) proposed by Freeman et al. [18] 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. Random Walk Betweenness-Centrality (RWBC), proposed by Newman [25], assumes that shorter paths are used 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 paper 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 node 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 which are not necessarily shortest [1, 23]. 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 paper 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, TLC, and FBC are particular cases of RBC. We elaborate on SPBC, TLC, and FBC in Section 2 and show how to define a routing strategy that will match their communication model in Section 3. We also add the notion of sampling rate which was not considered by prior algorithms for computing Betweenness-Centrality measures. We present a set of algorithms for computing RBC of individual nodes, sequences of nodes (e.g. links), and sets of nodes and/or links. Table 1 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 nodes 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 nodes. Conclusions appear in Section 6. Symbols and notation principles used in this paper are summarized in the Appendix.

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 [2, 16]. SPBC is defined as the sum of fractions of all shortest paths between each pair of nodes in a network which 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 node sends one packet to every other node. In this case, SPBC of a node v is the expected number of packets that traverse v. Brandes has shown in [10] that shortest paths from a single source s to all other nodes can be efficiently aggregated by traversing the nodes in the order of a non-increasing distance from s. Efficient aggregation of shortest paths is used to compute SPBC of all nodes in a network in $O(|V||E|)$ time, where V is the set of nodes and $|E|$ is the set of links in the network. SPBC can naturally be extended to Group Betweenness-Centrality (GBC) [14]. GBC of a set of nodes M is defined as the sum of fractions of all shortest paths which traverse at least one node in M. GBC of a single group can be computed in $O(|V||E|)$ [11, 30] or in $O(|M|^3)$ time following a preprocessing that takes $O(|V|^3)$ time.

The definition of GBC resembles the definition of the effectiveness of a group of distributed network monitors which is defined as the probability that a random packet is sampled at least once by at least one of the monitors [12]. Moreover, Holme has shown in [212], that the SPBC of a node is highly correlated with the fraction of time that the node is occupied by traffic. SPBC was also used by Yan et al. [34] 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 (TLC) assumes a more realistic routing strategy, where every node forewords the packet to a neighbor chosen randomly out of the neighbors that are closest to the target. like SPBC, TLC can be computed in $O(|V||E|)$ time for all nodes in the network [11, 24]. The group variant of TLC can also be computed in $O(|V||E|)$ as was indicated in [11]. To the best of our knowledge, there are no algorithms that reduce the time required to compute TLC of a group of nodes using preprocessing. SPBC and TLC possess similar statistical properties, however, normalized SPBC and TLC can differ up to 30% for individual nodes in large networks [36]. The drawback of SPBC and TLC 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. [18] introduce Maximal-Flow Betweenness-Centrality (FBC) that equally considers all paths from source to target. Roughly speaking, FBC of the node v is the sum of fractions of the maximal flows betweens each pair of nodes, that is transferred by the node 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 node 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 [8] 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 TLC. We present algorithms for computing RBC of individual nodes as well as sets and sequences of nodes. 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 TLC (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 paper 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 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 (for example, by analyzing historical behavior of the network).

Formally, let $G = (V, E)$ be a communication network topology where V is a set of n nodes and E is a set of links between the nodes. We do not allow self loops such as $(v, v) \in E$. Let T be a traffic matrix where $T(s, t)$ is the number of packets sent from a source node s to a target node 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.

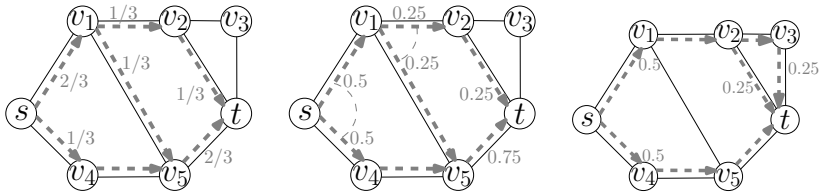
Assume, for example, that a group of monitors is installed on nodes in a network. The total number of bytes or the total importance of the communication passing through this group can be regarded as its monitoring potential. However the actual volume of information being monitored depends on the sampling 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.

TLC nodes 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 nodes forward packets to one of the neighbors which are closest to the target. The probability of



(a) SPBC (b) TLC (c) FBC

Figure 1: Sample network with traffic flowing from s to t according to SPBC, TLC, and FBC flow models.

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 nodes 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

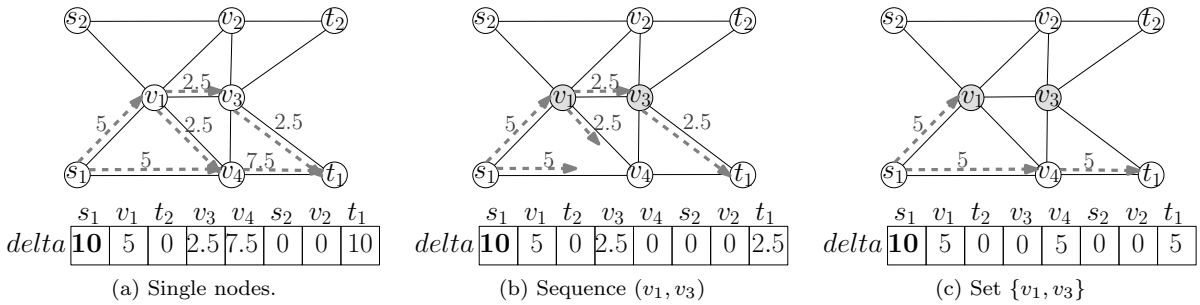
In this section we define Routing Betweenness-Centrality (RBC), focusing on routing schemes where the routing decisions depend on the source and the target of a packet. In Section 5 we will show how the computation of RBC can be optimized when the routing decisions are source-oblivious. In the next three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

4.1 RBC of individual nodes

Assume that a packet is introduced to the network by source node s and destined to leave the network at target node t. Let $\delta_{s,t}(v)$ be the probability that this packet will pass through the node 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.

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

Since we assume loop-free routing, $Pred_{s,t}$ defines a directed acyclic graph (DAG) [20] as shown in Figure 2a. 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 Equation 1 on all nodes starting from s.



(a) Single nodes. (b) Sequence (v_1, v_3) (c) Set $\{v_1, v_3\}$

Figure 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 nodes to their successors in Algorithms 1, 2, and 3 respectively.

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

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

$\delta_{\bullet,\bullet}(v)$ can be regarded as the potential of v to inspect or alter communications in the network. Equation 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 nodes in $O(n^2m)$ time using Equations 1 and 2.

Algorithm 1: RBC of nodes

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')
    $\{s = v_0 \preceq v_1 \preceq \ldots \preceq v_n = t\} \leftarrow$
    topologically sorted nodes 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 $v_j \in$ *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 RB

Algorithm 1 is composed of an outer loop that iterates over all s-t pairs of nodes 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 each one of the nodes is computed and these probabilities are accumulated according to Equation 2 to form RBC values of all nodes. Most of the following algorithms will use the same template and similar content.

4.2 RBC of ordered sequences

In this subsection we define RBC of ordered sequences of nodes. 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 nodes 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 in [28] as a technique to speed up the computation of shortest-path group Betweenness-Centrality (GBC) in an order of magnitude. We will also use the concept of sequence Betweenness-Centrality to speed up the computation of RBC of sets in Section 5.3.

Let $S = (s_1, \ldots, s_k)$ be a sequence of nodes. Let $\tilde{\delta}_{s,t}(S)$ be the probability that a single packet emanating from s and targeted at t will pass through all nodes 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 nodes. If the same node appears more than once, all successive appearances of the node 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 node in the sequence S are separated by a different node 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 nodes, for example $\tilde{\delta}_{s,t}((v,\ldots,s,\ldots)) = 0$. The following set of equations recursively computes the probability that a packet will pass through the sequence S:

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

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

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

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

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

4.3 RBC of sets

In this subsection we define the set variant of RBC. Generally, Betweenness-Centrality of a group of nodes measures the extent to which packets traverse at least one of the nodes in the group. The concept of centrality was first applied to groups and classes of nodes in networks by Everett and Borgatti in [14]. The set variant of RBC can be used, for example, for estimating the expected effectiveness of distributed monitors.

Algorithm 2: RBC of sequences (with sampling)

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

Algorithm 3: RBC of sets (with sampling)

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

Let $M = \{v_0, \ldots, v_k\}$ be a set of nodes. Let $\ddot{\delta}_{s,t}(M)$ be the probability that a packet from s to t will pass through at least one of the nodes 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 ρ_v be the sampling rate of the monitor installed on the node v. Let $M_\rho = \{v | \rho_v > 0\}$ be a set of nodes with positive sampling rates. M_ρ can be regarded as a fuzzy set where ρ_v is the extent to which v belongs to M_ρ. 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 nodes 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 market 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 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 Equation 5.

$$\begin{aligned} \lambda_{s,t}^{M_\rho}(s) &= (1 - \rho_s) \\ \lambda_{s,t}^{M_\rho}(v) &= (1 - \rho_v) \cdot \sum_{u \in Pred_{s,t}(v)} \lambda_{s,t}^{M_\rho}(u) \cdot R(s,u,v,t) \end{aligned} \tag{5}$$

$\lambda_{s,t}^{M_\rho}(t)$ 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:

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

The RBC of the fuzzy set M_ρ is the expected number of packets sampled by at least one node in M_ρ and can be computed using the inverse probabilities as described in Equation 6.

$$\ddot{\delta}_{\bullet,\bullet}(M_\rho) = \sum_{s,t \in V} (1 - \lambda_{s,t}^{M_\rho}(t)) \cdot T(s,t) \tag{6}$$

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

Equations 5 and 6 can be used to compute RBC of one group of monitors with given sampling rates in $O(n^2 m)$ as shown in Algorithm 3. In the input to Algorithm 3 we use ρ to represent nodes with positive sampling rates. In the propagation stage of Algorithm 3 only the 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 Equation 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 nodes in the networks, the algorithm computes the total expected number of packets that were captured by at least one monitor according to Equation 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 $O(n^2 m)$ 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 nodes, sets, and sequences and present their respective algorithms with minimal changes.

5.1 RBC of individual nodes

Let $\delta_{\bullet,t}(r)$ be the expected number of packets targeted at t that pass through the node r as defined by Equation 7.

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

$\delta_{\bullet,t}(r)$ 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 subsections we will show how to compute RBC of individual nodes, sequences, and sets by aggregating target dependencies. Since target dependency is a summation of pairwise dependencies over all sources, RBC of the node r is a summation of target dependencies over all targets as shown in Equation 8.

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

If we are able to compute target dependency directly without using Equation 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 nodes 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.

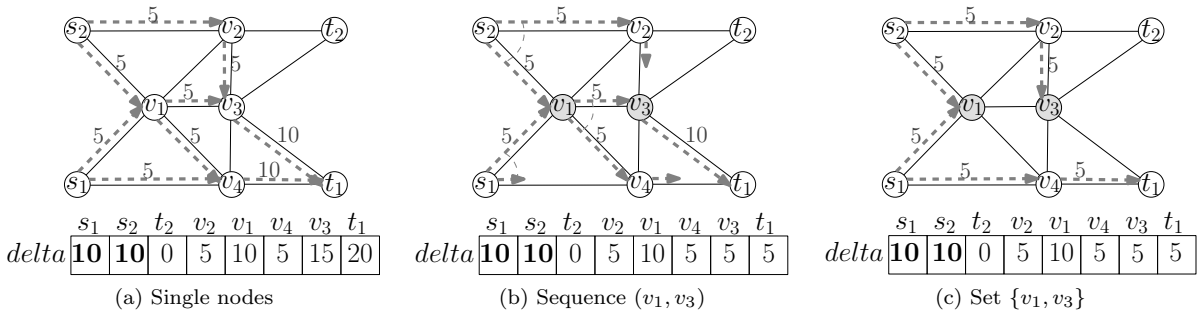
Let $Pred_t(v)$ be a set of all predecessors of v on the way to t: $Pred_t(v) = \{u | R(\oslash, u, v, t) > 0\}$. In contrast to $Pred_{s,t}(v)$ defined in Section 4, here the set of the possible predecessors of v is not influenced by the source of communication. Let u be a predecessor of v on the way to t. The probability of a packet to pass through v after visiting u is $R(\oslash, u, v, t)$. 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 by Equation 9.

$$\delta_{\bullet,t}(v) = T(v,t) + \sum_{u \in Pred_t(v)} \delta_{\bullet,t}(u) \cdot R(\oslash, u, v, t) \tag{9}$$

This equation can be derived directly from Equations 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 $Pred_t$ defines a DAG similarly to $Pred_{s,t}$, but this time the DAG has multiple sources and a single sink t as shown in Figure 3. Equation 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 Equations 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 nodes, we need to loop only through all target nodes to compute RBC given the source-oblivious routing strategy. Algorithm 4 loops once through all target nodes $t \in V$, 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 to the single source and single sink DAG, built by the algorithms in the previous section), sorting its nodes. 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 nodes of the network and aggregates



(a) Single nodes (b) Sequence (v_1, v_3) (c) Set $\{v_1, v_3\}$

Figure 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 nodes to their successors in Algorithms 1, 2, and 3 respectively.

Algorithm 4: s-oblivious RBC of nodes

Input: $G(V, E), R, T$
Output: $RBC[1..|V|]$
Data: $delta[1..|V|]$
$\forall_{v \in V}, RBC[v] = 0$
for $t \in V$ **do**
  ▼ **topological sort**
    $E' = \{(u, v) | R(\oslash, u, v, t) > 0\}$
    $D =$ directed acyclic graph (V, E')
    $\{v_0 \preceq v_1 \preceq \ldots \preceq v_n = t\} \leftarrow$
    topologically sorted nodes of D;
  ▼ **init delta**
    $\forall_{v \in V}, delta[v] = T(v, t)$;
  ► **accumulate** $\delta_{\bullet,\bullet}(v)$;
return RB

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 nodes 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 $O(n^2 m)$. 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 $S_\rho = (s_1, \ldots, s_k)$ be a sequence of nodes with sampling rates $\rho_{s_1}, \ldots, \rho_{s_k}$ respectively. Let $\tilde{\delta}_{\bullet,t}(S)$ be the expected number of packets targeted at t that pass through all nodes 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 nodes in the sequence. Equations 10 and 11 describe RBC of the sequence S_ρ in terms of $\tilde{\delta}_{\bullet,t}(S_\rho)$.

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

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

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

$$= \sum_{u \in Pred_t(v_k)} \tilde{\delta}_{\bullet,t}((\ldots, v_{k-1}, u)) \cdot R(\oslash, u, v_k, t)$$

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

Algorithm 5: s-oblivious RBC of sequences (with sampling)

```
Input: G(V,E), R, T, ρ, S = {s_0, ..., s_k}
Output: RBC
Data: delta[1..|V|]
RBC = 0
for t ∈ V do
    ► topological sort
    ▼ init delta
        for v ∈ V do
            if v ≺ s_0 or v is s_0 then
                delta[v] = T(v,t);
            else
                delta[v] = 0;
    ► accumulate δ̃_{•,•}(S_ρ);
return RBC
```

Algorithm 5 computes RBC of a sequence of monitors in $O(nm)$ time using Equations 10 and 11. During the iteration over all target nodes this algorithm sorts nodes, 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 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 nodes to the sequence (u,v) using the following lemma:

Lemma 1 (Dependency chaining) *Let $S = (s_1, \ldots, s_k)$ be an ordered sequence of nodes. The probability that a packet sent from a node s to a different node t will pass through all nodes in S in a given order is:*

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

Proof: *The following proof is based on the fact that the probability of a packet passing through $(s_2, \ldots, s_k)$, 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 $\tilde{\delta}_{s,t}((s_1, \ldots, s_k)) = 0$. $\tilde{\delta}_{s,t}((s_1, \ldots, s_k))$ 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 $(s_1, \ldots, s_k)$ 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 nodes $s_2, \ldots, s_k$.

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

Before we continue the proof for a more general case $\tilde{\delta}_{s,t}((s_1, \ldots, s_k)) \neq 0$ we will now show that for any set of nodes there is at most one permutation L of these nodes 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 nodes in the network. Let $M \subseteq V$ be a subset of nodes. 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 $L_1 = (v_1, \ldots, v_l)$ and $L_2 = (u_1, \ldots, u_l)$ 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 node v_i in L_2 ($v_i = u_j$). Let k be the index of the node u_i in L_1 ($u_i = v_k$). Since all nodes appear only once in both permutations and i is the lowest index for which nodes 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 nodes in the order defined by the permutation. In particular it holds that there is at least one route from s to t passing through $(v_i, \ldots, v_k)$ and similarly for $(u_i, \ldots, u_j)$. 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 $(u_i, \ldots, u_j = v_i)$ and continue back to $u_i = v_k$ through $(v_i, \ldots, v_k = u_i)$ in contradiction to the assumption that the routing strategy is loop-free.* ■

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

Assume that $\tilde{\delta}_{s,t}((s_1, \ldots, s_k)) > 0$. Let the event Z_v represent all cases where the packet passes through node 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.

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

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

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

Since we are dealing with source-oblivious routing the nodes that the packet passed prior to passing through s_1 (in particular the source node 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 $s_2, \ldots, s_k$ after visiting s_1.

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

Finally, according to the definitions of δ and $\tilde{\delta}$, the proof of Lemma 1 can be completed.

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

■

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

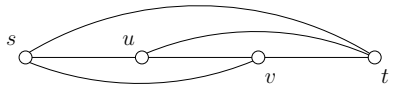


Figure 4: In this figure assume that packets are sent from s to t and are forwarded by u and v from the left to the right. The probability that an arbitrary packet sent from s to t will pass through u and v is smaller than the probability that it will pass through u. $\delta_{s,t}(u) = \frac{1}{3}$, $\delta_{s,t}(v) = \frac{1}{2}$, $\delta_{s,v}(u) = \frac{1}{2}$, 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.

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

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

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

Equations 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 nodes:

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

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 nodes 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 nodes in M_ρ:

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

Algorithm 6: s-oblivious RBC of sequences (with sampling, after preprocessing)

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

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

$$\lambda^{M_\rho}_{\bullet,t}(v) = (1-\rho_v)\cdot T(v,t) + \sum_{u \in Pred_t(v)} \lambda^{M_\rho}_{\bullet,t}(u) \cdot R(\oslash, u, v, t) \cdot (1-\rho_v) \tag{18}$$

$$\ddot{\delta}_{\bullet,\bullet}(M_\rho) = \sum_{t \in V} \left(\sum_{s \in V} T(s,t) - \lambda^{M_\rho}_{\bullet,t}(t) \right). \tag{19}$$

Algorithm 7 computes RBC of a set of monitors installed on nodes 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 nodes in the network and in each iteration, sorts nodes is 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.

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** $\ddot{\delta}_{\bullet,\bullet}(M_\rho)$;
return RBC

Contribution to RBC of a set. In this subsection we assume that a set of monitors X is installed on nodes in a network and their sampling rates are specified by ρ. 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 nodes, sets of nodes, or sequences of nodes to RBC of X_ρ. In Section 4.3 we have defined $\lambda^{X_\rho}_{s,t}(v)$ as the probability that a packet from s to t will pass through v without being sampled by monitors in X_ρ. This probability gives no information regarding the probability that this packet will be sampled after passing through v.

Let $\chi^{X_\rho}_{s,t}(w)$ 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_ρ (neither before nor after visiting w). Assume that v monitors the traffic with sampling rate $\rho_w > 0$. Then $\chi^{X_\rho}_{s,t}(w) \cdot \rho_v \cdot T(s,t)$ is the expected number of packets from s to t that were sampled only by w and not by other monitors. In other words, $\chi^{X_\rho}_{s,t}(w) \cdot \rho_w$ 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 nodes to X one at a time using the following lemma:

Lemma 2 (Pairwise dependency contribution) *Let $X = \{v_1, \ldots, v_k\}$ be a set of nodes with sampling rates specified by $\rho_{v_1}, \ldots, \rho_{v_k}$ respectively. Let w be a node with sampling rate ρ_w. For any $u \in V$ it holds that:*

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

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 node in X ($\tilde{\chi}_{s,t}^{X \cup \{w\}}(w)$), equals its probability of passing through w without being sampled by any node in X and *not being sampled by w. Being sampled by w and being sampled by any node 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 node v. Let the event Z_v represent the cases where this packet was sampled by the node v. Let $\overline{Z}_v$ represent the cases where this packet was not sampled by the node v. $\chi_{s,t}^{X_\rho}(u)$ is the probability that a packet from s to t was not sampled by any node in X, but passes through u:

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

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

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

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

According to 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 δ to χ by excluding packets that are sampled by some node in X.

By replacing the probabilities in Equation 22 with the respective pairwise dependencies we get:

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

According to the Dependency Chaining Lemma, which can also be adjusted to χ 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.

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

■

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_ρ. Lemma 2 can be used to compute $\chi_{\bullet,t}^{X_\rho \cup \{w\}_\rho}(u)$ by multiplying Equation 20 by $T(s,t)$ and summing it over all sources.

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

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 node in X. $\chi_{\bullet,\bullet}^{X_\rho}(w) \cdot \rho_w$ can be considered as the contribution of w to RBC of X_ρ:

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

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 $R(\oslash, u, v, t)$ 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 $M = \{v_0, \ldots, v_k\}$ be a set of nodes with sampling rates specified by $\rho_{v_0}, \ldots, \rho_{v_k}$ respectively. Let $M^{(i)} = \{v_0, \ldots, v_i\}$ 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 node v. Let the event $\overline{Z_v}$ represent all cases where the packet from s to t is not sampled by the node v. Let the event Y_v represent all cases where the packet from s to t passes through node v. By definition, $\ddot{\delta}_{s,t}(\{v_1, \ldots, v_k\})$ is the probability that the packet, from s to t, will pass through at least one of the k nodes $v_1, \ldots, v_k$

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

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

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

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 $v_1,\ldots,v_{i-1}$. According to the definition of χ

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

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

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

■

Algorithm 8: s-oblivious RBC of sets (with sampling, after preprocessing)

Input: $G(V,E)$, R, T, ρ, $M=\{v_0,\ldots,v_k\}$, $\delta_{s,t}(v)$, $\delta_{\bullet,t}(v)$
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

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

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

Summing Equation 26 over all targets results in iterative computation of RBC of a set of nodes:

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

In the last algorithm presented in this paper (Algorithm 8), we compute RBC of a given set iterating over all nodes in the set and summing their marginal contributions as described in Equation 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 Equation 20 to update pairwise dependencies and Equation 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 nodes. Temporal arrays *pdep* and *tdep* maintain pairwise and target dependencies respectively. Initially the values in these arrays correspond to $\chi_{s,t}^{\{\}}(v) = \delta_{s,t}(v)$ and $\chi_{\bullet,t}^{\{\}}(v) = \delta_{\bullet,t}(v)$. In each iteration of the outer loop we process one node from the input set M. The marginal contributions of nodes to RBC of M are aggregated according to Equation 27 during the first inner loop (that iterates over all $t \in V$). The second and third inner loops iterate over nodes in M and update the entries of the *tdep* and *pdep* matrices respectively according to Equations 25 and 20. After the first update of all the values in these matrices they correspond to $\chi_{s,t}^{M_\rho^{(1)}}(v)$ and $\chi_{\bullet,t}^{M_\rho^{(1)}}(v)$ where $M^{(1)}$ contains only the first node in M. During the second iteration we process one more node from M and update the matrices again and so on until we process all nodes in M. The overall time complexity of Algorithm 6 is $O(k^3 n)$ where k is the size of M and n is the number of nodes 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 nodes. The problem of monitoring links can easily be reduced to a problem of monitoring nodes by adding a phantom node 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 nodes in their main loop. This is an intuitive optimization since they do not introduce traffic to the network. In this case, adding phantom nodes 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 nodes, the size of the pre-computed matrices will depend on the number of edges and not only the number of nodes. Thus, adding phantom nodes increases both time and space complexity of the preprocessing stage in Algorithms 6 and 6 from $O(n^2 m)$ time and $O(n^3)$ space to $O(|E|^2 m)$ time and $O(|E|^3)$ space.

In order to avoid the addition of phantom nodes we should remember that the RBC of a (directed) link is the RBC of the sequence consisting of both of its' nodes. Therefore, the RBC of a sequence of directed links is RBC of the sequence of nodes 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 nodes.

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

Algorithm 9: s-oblivious RBC of sets of links (with sampling, after preprocessing)

Input: $G(V,E), R, T, \rho, Q = \{(u_1,v_1),\ldots,(u_k,v_k)\}$, $\delta_{s,t}(v)$, $\delta_{\bullet,t}(v)$
Output: RBC
Data: pdep, npdep, tdep, ntdep
$RBC = 0$
X = set of nodes 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$[u,t]$·pdep$[u,v,t]$
      for $w \in X$ **do**
        ntdep$[w,t]$ = tdep$[w,t]-$
          $-$tdep$[w,t]$· pdep$[w,u,t]$· pdep$[u,v,t]-$
          $-$tdep$[w,t]$· pdep$[w,u,t]$· pdep$[u,v,t]$
        for $s \in X$ **do**
          npdep$[s,w,t]$ = pdep$[s,w,t]-$
            $-$pdep$[s,w,t]$· pdep$[w,u,t]$· pdep$[u,v,t]-$
            $-$pdep$[s,w,t]$· pdep$[w,u,t]$· pdep$[u,v,t]$;
  tdep=ntdep; pdep=npdep;
return RBC

of the sequence (u,w,v) is zero. We can modify the recursive Equation 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 $Q^{(i)} = \{(u_1,v_1),\ldots,(u_i,v_i)\}$ be a set of nodes with sampling rates specified by $\rho_{(v_1,u_1)},\ldots,\rho_{(u_i,v_i)}$ respectively.

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

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

Algorithm 10: s-oblivious RBC of sets of links and nodes (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 nodes 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

Algorithm 9 computes the RBC of a set of links similarly to the way that Algorithm 8 computes the RBC of a set of nodes. But instead of implementing the Equations 20 and 25 it implements the Equation

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 of nodes attached to the input links.

Algorithm 9 has the same four nested loops as Algorithm 8. In addition the number of nodes 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 nodes and monitors installed on links as shown in Algorithm 10. Let M be the set of monitored nodes and Q be the set of monitored links. Let $X = M \cup \{u, v : (u, v) \in Q\}$ be the set of nodes comprising M and Q. First all the data relevant to X is copied into the *tdep* and *pdep* matrices. Afterwords, the cores of both algorithms are executed consequently to compute RBC of the mixed set $M \cup Q$.

6 Conclusions

In this paper 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 nodes or groups of nodes are exposed to the traffic given any loop-free routing strategy.

The algorithms presented in this paper are easily modified to compute RBC of groups consisting of links and/or nodes (see Appendix 5.4). In fact a more sophisticated combinations of policies for traffic monitoring/controlling are supported. Using the methods present in this paper 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 prepossessing can dramatically reduce the time required for a single computation of RBC of a sequence. Prepossessing 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).

Since RBC is more general than existing known definitions of Betweenness-Centrality and capable of better reflecting routing schemes in communication networks we believe that many applications of RBC will be found in the near future. Currently we have already found RBC 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 such as those described in [13, 29, 32] for optimizing placement of passive monitors within the communication network. Other obvious applications include simulation free prediction of congestions in communication networks, design and examination of routing strategies and network layout, for, say, balancing the traffic load in the network and assuring service level agreements.

Acknowledgments

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

[1] Optimized multipath. http://www.faster-light.net/omp/.

[2] J. M. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, 1971.

[3] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] A.-L. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A*, 281:69–77, 2000.

[5] M. Barthélemy. Betweenness centrality in large complex networks. *The European Physical Journal B – Condensed Matter*, 38(2):163–168, March 2004.

[6] B. Bollobas and O. Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics*, 1(1):1–35, 2003.

[7] S. P. Borgatti. Centrality and network flow. *Social Networks*, 27:55–71, 2005.

[8] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, Oct. 2006.

[9] P. Bork, L. J. Jensen, C. von Mering, A. K. Ramani, I. Lee, and E. M. Marcotte. Protein interaction networks from yeast to human. *Curr. Opin. Struct. Biol.*, 14(3):292–299, Jun. 2004.

[10] U. Brandes. A faster algorithm for betweenness centrality. *Mathematical Sociology*, 25(2):163–177, 2001.

[11] U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, 2008.

[12] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran. Reformulating the monitor placement problem: optimal network-wide sampling. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2006. ACM.

[13] S. Dolev, Y. Elovici, R. Puzis, and P. Zilberman. Incremental deployment of network monitors based on group betweenness centrality. *to appear in Information Processing Letters.*

[14] M. G. Everett and S. P. Borgatti. The centrality of groups and classes. *Mathematical Sociology*, 23(3):181–201, 1999.

[15] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Comm. Rev.*, 29(4):251–262, 1999.

[16] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[17] L. C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1:215–239, 1979.

[18] L. C. Freeman, S. P. Borgatti, and D. R. White. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141–154, Jun. 1991.

[19] K.-I. Goh, B. Kahng, and D. Kim. Universal behavior of load distribution in scale-free networks. *Phys. Rev. Lett.*, 87(27):278701, Dec. 2001.

[20] F. Harary, R.Z. Norman, and D. Cartwright. *Structural models. An introduction to the theory of directed graphs.* John Wiley and Sons, New York, 1965.

[212] P. Holme. Congestion and centrality in traffic flow on complex networks. *Advances in Complex Systems*, 6(2):163–176, 2003.

[22] A.W. Jackson, W. Milliken, C.A. Santivanez, M. Condell, and W.T. Strayer. A topological analysis of monitor placement. *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*, pages 169–178, July 2007.

[23] J. Moy. Rfc 2328 - osfp version 2. http://www.ietf.org/rfc/rfc2328.txt, Apr. 1998.

[24] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132, 2001.

[25] M. E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39–54, Jan. 2005.

[26] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Phys. Rev. E*, 65:036104, 2002.

[27] S. Porta, P. Crucitti, and V. Latora. The network analysis of urban streets: a primal approach. *Environment and Planning B: Planning and Design*, 33(5):705–725, September 2006.

[28] R. Puzis, Y. Elovici, and S. Dolev. Fast algorithm for successive computation of group betweenness centrality. *Phys. Rev. E*, 76(5):056709, 2007.

[29] R. Puzis, Y. Elovici, and S. Dolev. Finding the most prominent group in complex networks. *AI Comm.*, 20:287–296, 2007.

[30] R. Puzis, D. Yagil, Y. Elovici, and D. Braha. Collaborative attack on internet users' anonymity. *Internet Research*, 19(1):60–77.

[31] S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, March 2001.

[32] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: Complexity, heuristics, and coverage. *Computer Communications*, 29:1564–1577, 2006.

[33] S. Wasserman and K. Faust. *Social network analysis: Methods and applications.* Cambridge, England: Cambridge University Press., 1994.

[34] G. Yan, T. Zhou, B. Hu, Z.-Q. Fu, and B.-H. Wang. Efficient routing on complex networks. *Phys. Rev. E*, 73:046108, 2006.

[35] S.-H. Yook, H. Jeong, and A.-L. Barabasi. Modeling the internet's large-scale topology. *Proceedings of the National Academy of Science*, 99(21):13382–13386, Oct. 2002.

[36] T. Zhou, J.-G. Liu, and B.-H. Wang. Notes on the algorithm for calculating betweenness. *Chinese Physics Letter*, 23:2327–2329, Aug. 2006.

Table 2: Notations

i, j, k, l	Natural numbers – indexes or sizes of sets and sequences.
n	The number of nodes in the network.
m	The maximal number of edges in the routing tree (or the routing DAG for multi-path routing schemes).
r, s, t, u, v, w, x, y, z	Nodes.
$G(V, E)$	A network with nodes V and edges E.
M, X	Sets of nodes.
S	Sequence of nodes.
ρ_v	Sampling rate of v.
M_ρ, X_ρ, S_ρ	Sets and sequences where the sampling rate of members is specified by ρ.
$\delta_{s,t}(v)$, $\ddot{\delta}_{s,t}(M)$, $\tilde{\delta}_{s,t}(S)$	Pairwise dependency of s and t on a node v, a set M, or a sequence S respectively.
$\delta_{\bullet,t}(v)$, $\ddot{\delta}_{\bullet,t}(M)$, $\tilde{\delta}_{\bullet,t}(S)$	Target dependency of t on a node v, a set M, or a sequence S respectively.
$\delta_{\bullet,\bullet}(v)$, $\ddot{\delta}_{\bullet,\bullet}(M)$, $\tilde{\delta}_{\bullet,\bullet}(S)$	The Betweenness-Centrality of a node v, a set M, or a sequence S respectively.
$\lambda_{s,t}^{X_\rho}(v)$, $\lambda_{\bullet,t}^{X_\rho}(v)$	Same as $\delta_{s,t}(v)$ and $\delta_{\bullet,t}(v)$ but accounts only for packets not sampled by the monitors in X prior to reaching v.
χ^{X_ρ}	Same as δ but accounts only for packets not sampled by the monitors in X.
p, Pr	Probability.
$\sigma_{s,t}$, $\sigma_{s,t}(v)$	Number of shortest paths between s and t and the number of them that pass through v (used for computing SPBC).
$\phi_{s,t}$, $\phi_{s,t}(v)$	Maximal flow between s and t and flow transferred by v respectively (used for computing FBC).

APPENDIX A. Notations

Next we will summarize the symbols and the notation principles used in this paper (see Table 2). The uppercase Latin letter G represents a network. Other uppercase Latin letters are used to represent sets and sequences. For example, V is the set of nodes in G, $M \subset V$ is an arbitrary set of nodes in G, and S is an arbitrary sequence of nodes. E denotes the set of edges in G. Lowercase Latin letter n denotes the number of nodes in the network. The letter m denotes the number of edges in a routing DAG rooted at some target vertex. When m is used to specify the complexity of an algorithm it represents the maximal number of edges in any routing DAG. Natural numbers and indexes of arrays and sequences are denoted by i, j, k, or l. The letters r, s, t, u, v, w, x, y, and z represent nodes. p and Pr are used to represent a probability. $R(s, u, v, t)$ is a quaternary function that encodes the routing scheme in the network.

Greek letters represent an influence on the traffic. ρ is used to denote the sampling rate of monitors. In previous subsections we defined several *pairwise dependencies* of a pair of source-target nodes on another node ($\delta_{s,t}(v)$), set of nodes ($\ddot{\delta}_{s,t}(M)$), or sequence of nodes ($\tilde{\delta}_{s,t}(S)$). In general the double-dot accent $\ddot{}$ is added to functions related to RBC of sets and the tilde accent $\tilde{}$ is added to functions related to RBC of sequences. λ, that was defined in Section 4.3, and χ, that will be defined in section 5.3, also represent *pairwise dependencies* like δ. However, λ refers to the probability of a packet not being sampled prior to reaching the argument node or nodes and χ refers to the probability of a packet not being sampled at all. We use a bullet ($\bullet$) with δ, λ, or χ instead of the subscript indexes to indicate that the *pairwise dependency* is summed over all sources and/or targets. RBC is denoted using bullets instead of both subscript indexes (s and $\delta_{\bullet,\bullet}(v)$). *Target dependency* will be denoted in the next section using a bullet instead of the first subscript index ($\delta_{\bullet,t}(v)$).