Identifying Bottleneck Nodes using Packet Delay Statistics

Joshua Marker, John M. Shea, and Tan F. Wong
University of Florida, Gainesville, FL 32606

Eric Graves and Paul L. Yu
Army Research Lab, Adelphi, MD 20783

***Abstract*—An algorithm to identify the bottleneck nodes linking two component networks in a simple network of networks (NoN) configuration is proposed. The proposed bottleneck identification algorithm is based on applying a support vector machine on clustered packet delay measurements. This algorithm has the advantage that it requires almost no information about the topology of the underlying NoN. Simulation results show that this algorithm can provide very good detection performance when the component networks of the NoN are not too small in size, or when the connectivity between nodes within the component networks is not too sparse.**

I. Introduction

Communication networks, such as the Internet, are often not homogeneous but are instead actually loosely interconnected sets of heterogeneous subnetworks. Such a "network of networks" [1] is characterized by difference in the connectivity within the subnetworks versus the connectivity between subnetworks. The links interconnecting subnetworks are referred to as *structural bottlenecks* in [1], as these are topological features of the network that restrict communication because traffic between subnetworks can only flow over the bottleneck links. These bottlenecks links and the network nodes attached to them are weak points in the network topology; disrupting these links/nodes can greatly alter the network topology and often cause the network to partition. Thus, techniques to identify bottlenecks and to obfuscate bottlenecks from adversaries are of significant interest.

Network connectivity is often studied by modeling the network as a graph, where the vertices represent the communication nodes and the edges represent the communication links. Then bottlenecks in the network can be found as edge cuts of the graph, where the number of edges cut is small but the subnetworks created by the cut do not have small order (vertex cardinality). The latter constraint is because most networks have some nodes that are considered to be near the perimeter of the network in that they are only connected to one or two other nodes, and the links connecting such nodes to the main network are not typically considered bottlenecks. When the topology of the network is known, then there are many algorithms for finding edge cuts of a graph [2]–[5]. A recent contribution [6] focuses particularly on bottleneck detection, and uses a linear program to find a bottleneck cut that mimics the cut associated with the Cheeger constant of the network [7].

This research was supported by the Army Research Laboratory under Grant W911NF-17-2-0182. J. M. Shea and T. F. Wong were also supported in part by National Science Foundation grant 1642973 and by AFOSR award number FA9550-19-1-0169.

In many situations, it may be desirable to detect a network bottleneck without knowing the network topology. For example, an adversary wishing to conduct a denial-of-service attack may not have access to the full network topology information. From the defense side, a network operator may wish to evaluate what features will make the network bottlenecks detectable by an adversary. For mobile networks, the bottlenecks may be time-varying based on the locations of the communicators, and a network operator may wish to characterize these bottlenecks, even if it does not know the topology of the full network. If the network allows conventional probing (for instance, using `traceroute` or Simple Network Management Protocol (SNMP)), then the topology can be easily discovered; however, network managers may disable these protocols to protect the details of their network. Techniques that use measurements of communications over a network to infer the topology of the network or the properties of the network links are referred to as network tomography techniques. Surveys of network tomography techniques are given in [8] and [9].

Most network tomography techniques for topology discovery are based on trying to infer a tree topology from a source to multiple receivers. In [10], techniques are developed to perform tomography from multiple sources to enable better characterization of network topology further from the source. However, the main result is to determine where two trees join in the progression from the source to the destinations in the logical network. This concept of detecting joining points is relevant to the problem of bottleneck identification because transmissions from multiple sources that must travel over a single bottleneck to reach another network must have a joining point (where the remaining routes are identical) at or before the bottleneck. The fact that the joining point may be far before the bottleneck limits the applicability of this technique to bottleneck detection.

In this paper, we consider a scenario in which network load results in different queuing delays at the different nodes in the network, and we wish to use the observed end-to-end delays to identify the location of a bottleneck between two subnetworks. We develop techniques to accurately identify a single bottleneck link that interconnects two subnetworks, while only using end-to-end delay measurements. The agents (observers) that are attempting to identify the bottleneck are constrained to be in the same subnetwork. We evaluate the

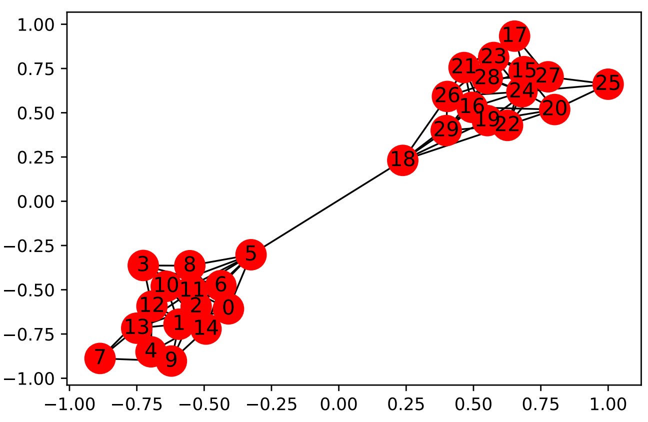

Fig. 1. A single-bottleneck graph with ten nodes in each component, representing a NoN with two networks connected by a single link.

performance for a scenario in which the observers can either hear round-trip traffic or can generate pings (and receive responses) at one or more nodes in the subnetwork. We apply clustering of the average delay measurements and use a fuzzy support vector machine classifier to identify the bottleneck nodes.

II. System Model

In this paper, we consider the simple network-of-networks (NoN) scenario in which two networks are connected together by a single link: the *bottleneck*. This NoN is represented by a *single-bottleneck graph*, which is defined as follows:

Definition 1. *We say that $G = (V, E)$[1] is a single-bottleneck graph if*

- *G is connected, and*
- *G contains a (unique) bridge $b \in E$ that cuts G into two bridgeless, connected components.*

The bridge $b = (u, v)$, where $u, v \in V$, models the bottleneck link joining the two network. Then the two connected components, produced from the cut by b, that contain u and v will be denoted by $G_u = (V_u, E_u)$ and $G_v = (V_v, E_v)$, respectively. Clearly, G_u and G_v model the two networks connected by b. We will call b as the bottleneck and u, v as the bottleneck nodes. We note the requirement that G_u and G_v are both bridgeless is to avoid the confusion of having multiple edges that may be considered as possible bottleneck links in G. Fig. 1 shows an example of a single-bottleneck graph with 10 nodes in each component. In Fig. 1, nodes 4 and 17 are the bottleneck nodes u and v, respectively. The bridge is the edge $b = (4, 17)$. Nodes 0–9 form the component network G_u, and nodes 10-19 form G_v.

Our goal is to develop an algorithm that an observer may be used to determine which two nodes among all the nodes in the NoN are the bottleneck nodes when minimal information about the network topology and traffic pattern is available. For example, the observer may not be able to easily obtain information about the topology of the network if Internet Control Message Protocol (ICMP) is disabled to protect against denial-of-service (DoS) attacks. More precisely, we make the following assumptions regarding the initial prior knowledge that the observer has about the network G and the observations that the observer can make to enhance his knowledge about G:

1) The observer knows that G is a single-bottleneck graph. He also knows all the nodes in G, i.e., the set V. Other than these two piece of information, the observer has no further information about the topology of G. In particular, he does not know u, v, E, V_u, or V_v.
2) The observer is able to send packets from one or more nodes, called *observer source nodes* (OSNs), to any other node in G, and measure the round-trip delay of each sent packet reaching the destination node and then the acknowledgment arriving back at the OSN.
3) If the observer uses multiple OSNs, then all the OSNs are located within the same component network (either G_u or G_v) of G, but the observer does not know to which component the OSNs belongs.
4) The observer does not know the underlying traffic pattern in G but does know that the networking delay at a node is dependent on the amount of traffic passing through that node.

Under these assumptions, the observer aims to identify the bottleneck nodes u and v amongst all nodes in V based only on the delay measurements obtained by sending packets from the OSNs to different destination nodes in G.

Because of the assumption that the observer may identify the bottleneck nodes based only on delay measurements, the pattern of network traffic over G can be sufficiently summarized by the distributions of the round-trip delay over the paths between any two nodes in G. Following the prior work described in Section I, we employ the following simple path-delay model. Assume that a minimum-hop (min-hop) routing scheme is applied to route packets between any two nodes in the network G. Henceforth, a min-hop path is implicitly assumed when we refer to a path between a pair of nodes in G. Assume that the queuing delay of a packet or its acknowledgment, going through the forward or return path from an OSN to a destination node, incurred at each node along the path is modeled by an exponential random variable with a mean that is proportional to the betweenness centrality (BC) of that node. The delay random variables incurred at all nodes along such a returned path are statistically independent, and the total round-trip delay measured by the OSN is the sum of the individual delays at all nodes traversed in the returned path. This simple delay model approximately represents the traffic scenario in which the network G is fully loaded in that there is a unit traffic flow between each pair of nodes in G. In this case, the BC of a node provides a reasonable measure that indicates the amount of traffic going through that node. Hence the queuing delay through the node is given by an exponential

[1]As usual, V and E denote the set of vertices (nodes) and the set of edges (links) of G, respectively.

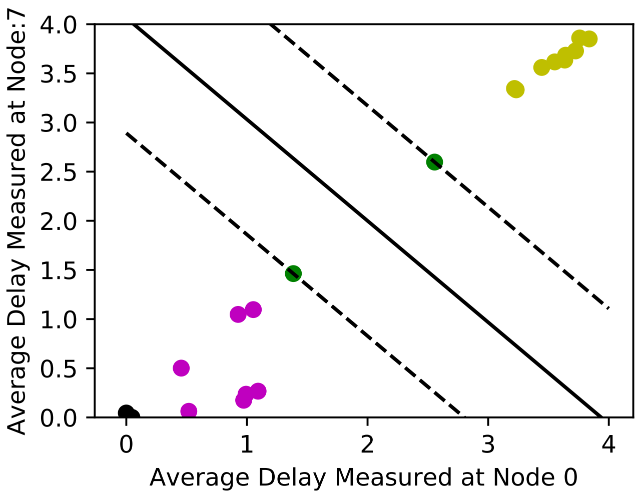

Fig. 2. A K-means clustering of the average delay vectors obtained for the network shown in Fig. 1. There are two OSNs, nodes 0 and 7, shown in black. Nodes classified to class 1 are in yellow, and the two OSNs belong to class 1. Nodes classified to class 2 are in magenta. The bottleneck nodes 4 and 17 (shown in green) are classified to classes 1 and 2, respectively.

random variable with a mean proportional to the BC for the case in which the service process is modeled by an M/M/1 queue.

III. Bottleneck Identification Algorithm

In this section, we propose an algorithm to identify the bottleneck nodes u and v amongst all nodes in V based only on the delay measurements obtained at the OSNs. The proposed bottleneck identification algorithm consists of the following three steps:

a) Preprocessing of delay measurements: Let $S \subseteq V$ be the set of OSNs. From assumption 3 in Section II, we have either $S \subseteq V_u$ or $S \subseteq V_v$. Consider each $s \in S$. Suppose that it sends $N_{s,v}$ packets to each $v \in V \setminus \{s\}$ and measures the average round-trip delay

$$D_{s,v} = \frac{1}{N_{s,v}} \sum_{i=1}^{N_{s,v}} D_{s,v}(i),$$

where $D_{s,v}(i)$ is the round-trip delay measured for the ith packet sent by s to v. For completeness, define $D_{s,s} = 0$. For each $v \in V$, form a $|S|$-dimensional average delay vector $D_v = (D_{s,v})_{s\in S}$ by aggregating the average round-trip delays from the OSNs in S to this node v. The collection $\mathcal{D} = \{D_v\}_{v\in V}$ of the average delay vectors is passed on to the next step.

b) Clustering of delay measurements: The $|S|$-dimensional average delay vectors in the collection $\mathcal{D}$ are then classified into two clusters (classes), $\mathcal{D}_1$ and $\mathcal{D}_2$ by a standard clustering algorithm. As an illustration, Fig. 2 shows the two clustered classes of the average delay vectors obtained for the network in Fig. 1. There are two OSNs, nodes 3 and 6, in this case. That is, $|S| = 2$, and there are twenty average delay vectors of dimension two in the collection $\mathcal{D}$. Ten thousand packets are randomly picked from an OSN to a destination node in V. The two OSNs are picked with equal probabilities, and then the destination node is picked from the other nodes in V with again equal probabilities. The collection of the twenty 2-dimensional average delay vectors are obtained from the 10000 round-trip delay measurements according to the preprocessing step a). Then the 20 average delay vectors are classified into two classes using the K-means algorithm. The figure shows that the two classes accurately identify the two component networks in G.

c) Bottleneck node identification: The clustering $\{\mathcal{D}_1, \mathcal{D}_2\}$ of $\mathcal{D}$ generates class labels for the nodes. For each $v \in V$, define the class label

$$y_v = \begin{cases} 1 & \text{if } D_v \in \mathcal{D}_1 \\ -1 & \text{if } D_v \in \mathcal{D}_2. \end{cases}$$

The following support vector machine (SVM) algorithm is then applied to the collection of the labeled average delay vectors $\{(y_v, D_v)\}_{v\in V}$:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w + c \sum_{v\in V} \xi_v \qquad (1) \\ \text{subject to} \quad & y_v \left(w^T \phi(D_v) + b \right) \geq 1 - \xi_v, \\ & \xi_v \geq 0, \ \text{ for each } v \in V \end{aligned}$$

where w is a $|S|$-dimensional linear weight vector, b is a scalar, and ϕ is the kernel function. The optimal SVM margin hyperplanes are then obtained. All nodes v with D_v lying in the parallelepiped bounded by and including the SVM margin hyperplanes are identified as bottleneck nodes. The regularization constant $c > 0$ in (1) controls the SVM fuzziness factor [11] that determines the bottleneck identification performance of our algorithm.

For example, the SVM margin hyperplanes (lines) of the collection of the labeled average delay vectors in Fig. 2 are shown by the broken lines, and the optimal SVM decision hyperplane is shown by the solid line in the figure. Note that $c = 10$ in this case. The two bottleneck nodes are the only nodes in the region bounded by the two SVM margin lines; hence they will be correctly identified as bottleneck nodes, and no non-bottleneck nodes are misidentified as bottleneck nodes.

IV. Simulation Results

In this section, we perform computer simulation to evaluate the performance of the bottleneck identification algorithm proposed in Section III. For all the results presented in this section, the NoN G is generated by the following procedure:

1) Generate an Erdős-Rényi graph $G_1 = (V_1, E_1)$ with K nodes and probability of edge creation p.
2) G_1 is checked to make sure that it is connected and bridgeless as required in Definition 1. If not, repeat step 1).
3) Randomly pick a node u amongst all nodes in V_1 with equal probabilities.

4) Repeat steps 1) – 3) to generate $G_2 = (V_2, E_2)$ and pick a node $v \in V_2$.
5) Form $G = (V, E)$ by letting $V = V_1 \cup V_2$ and $E = E_1 \cup E_2 \cup \{(u, v)\}$. That is, join G_1 and G_2 together to form G by adding the bottleneck edge between u and v. Rename G_1 and G_2 as G_u and G_v, respectively.
6) Pick $|S|$ nodes in V_u as OSNs uniformly.

After G is generated, the set of round-trip delays are generated according to the following procedure:

1) An OSN s is selected from all nodes in S with equal probabilities, and a v is selected from all nodes in $V \backslash \{s\}$ with equal probabilities.
2) The queuing delay each arrival of a probe packet at each node on the round-trip path from s to v is generated as an exponential random variable with mean proportional to the betweenness centrality of that node, where the delays are independent. The round-trip delay from s to v is found by adding up the set of delays along the returned path from s to v.
3) Steps 1) and 2) are repeated until a total of N round-trip delays have been calculated. That is, $N = \sum_{s \in S} \sum_{v \in V \backslash \{s\}} N_{s,v}$.

The algorithm described in Section III is then applied to identify the bottleneck nodes based on the set of round-trip delays.

We consider two metrics, namely a strong metric and a weak one, to evaluate the proposed bottleneck identification algorithm. For the strong metric, we define the *detection (true positive)* event be that **both** bottleneck nodes are identified as such and the *false alarm (false positive)* event be that **any** non-bottleneck nodes are identified as bottleneck nodes. On the other hand, for the weak metric, detection is defined as the event that a bottleneck node is identified as such and false alarm is defined as the event that a non-bottleneck node is identified as a bottleneck node. For each metric, we estimate the detection and false alarm probabilities from 5000 independent realizations for G. The estimated detection and false alarm probabilities obtained by varying the SVM fuzziness factor c in (1) are plotted to obtain a standard receiver operating characteristic (ROC) curve. Below, we will refer to the ROC curves based on the strong (resp., weak) metric as strong (resp., weak) ROC curves.

Fig. 3 shows the strong and weak ROC curves obtained for the cases in which each component network of G has 10 nodes (i.e., $|V_u| = |V_v| = 10$) and for probability of edge creation $p = 0.25, 0.5,$ and 0.75. The number of OSNs employed is $|S| = 2$. The total number of packets used to obtain delay measurements is $N = 2000$. We see from the figures that both the weak and strong ROC performance improves when the network components becomes more connected within themselves. When the network components are only lightly connected, the ROC performance is limited for this small network scenario. This result is intuitive. With the small network components, the traffic load across the bottleneck node is not much higher than that within each components. On the other hand, since the network components are lightly connected within themselves, the delay of a path within a network component may not be much lower than that of a path across the two component networks. All these makes it harder to identify the bottleneck nodes based on the delay measurements.

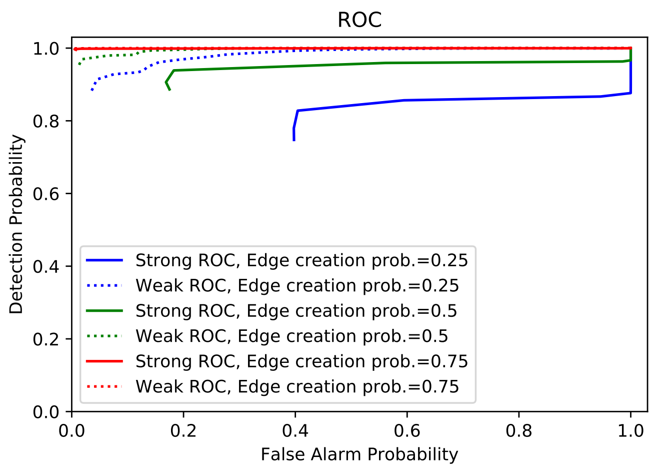

Fig. 3. ROC performance of NoNs with two components of ten nodes each. Number of OSNs $|S| = 2$. Total number of packets measured $N = 2000$.

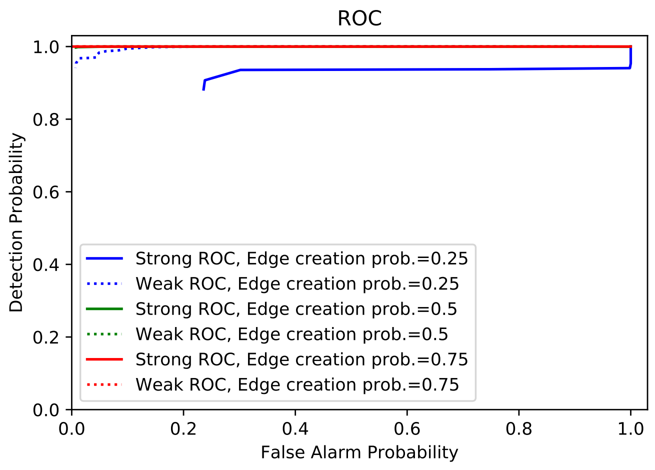

Fig. 4. ROC performance of NoNs with two 25-node components. Number of OSNs $|S| = 2$. Total number of packets measured $N = 5000$.

Fig. 4 shows the strong and weak ROC curves obtained for the cases in which each component network of G has 25 nodes (i.e., $|V_u| = |V_v| = 25$) and for probability of edge creation $p = 0.25, 0.5,$ and 0.75. The number of OSNs employed is $|S| = 2$. For a fair comparison with Fig. 3, the total number of packets used to obtain delay measurements is increased to $N = 5000$ in this case. Comparing Fig. 4 with Fig. 3, we observe that the ROC performance significantly improves for the larger network. This result is again intuitive in that the traffic load is heavier across the bottleneck when the network components become larger; hence it becomes easier to identify the bottleneck nodes based on the delay statistics.

In Figs. 5 and 6 we investigate the effect of the number

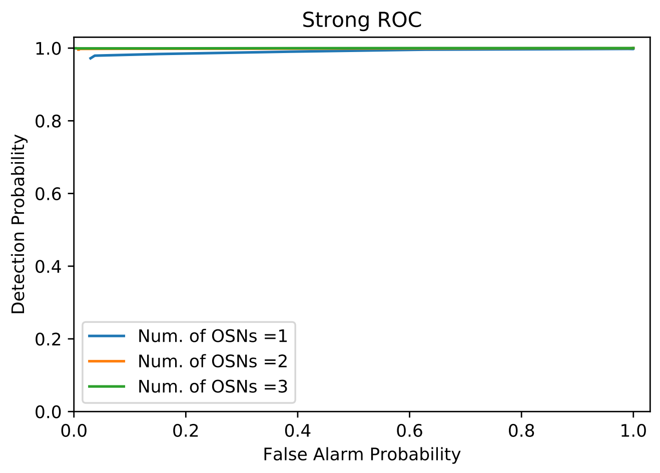

Fig. 5. Strong ROC performance of NoNs with two 25-node components with one, two, and three OSNs. Total number of packets measured $N = 5000$. Edge creation probability $p = 0.5$.

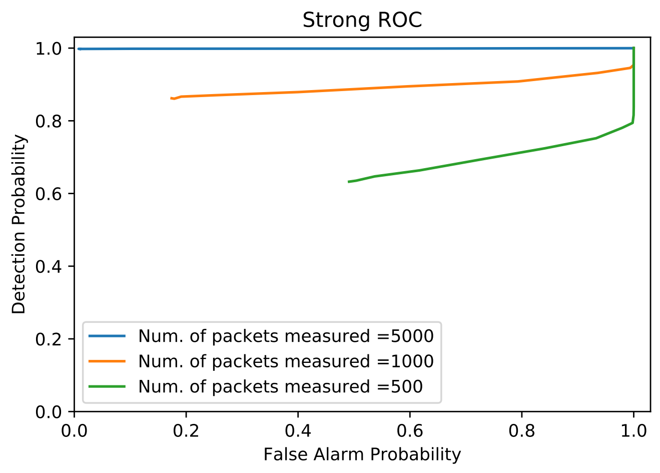

Fig. 7. Strong ROC performance of NoNs with two 25-node components with total number of packets measured $N = 5000, 1000$, and 500. Number of OSNs $|S| = 2$. Edge creation probability $p = 0.5$.

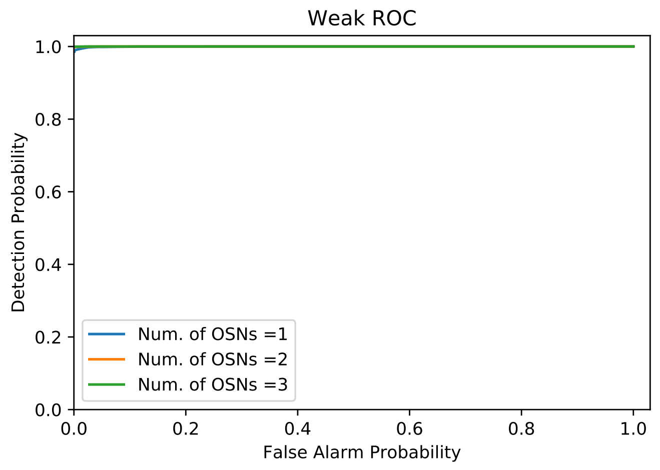

Fig. 6. Weak ROC performance of NoNs with two 25-node components with one, two, and three OSNs. Total number of packets measured $N = 5000$. Edge creation probability $p = 0.5$.

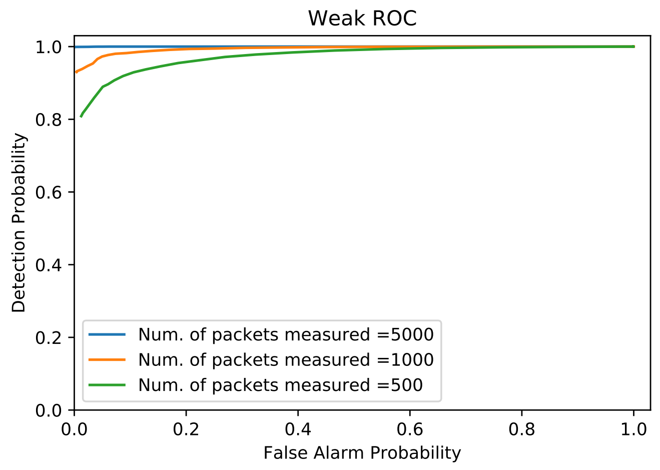

Fig. 8. Weak ROC performance of NoNs with two 25-node components with total number of packets measured $N = 5000, 1000$, and 500. Number of OSNs $|S| = 2$. Edge creation probability $p = 0.5$.

of OSNs on the bottleneck identification performance of the proposed algorithm. Plotted in the figures are the strong and weak ROC curves obtained for the case in which each component network of G has 25 nodes (i.e., $|V_u| = |V_v| = 25$), the probability of edge creation $p = 0.5$, and the number of OSN(s) $|S| = 1, 2$, and 3. The total number of packets used to obtain delay measurements is fixed to $N = 5000$ for all cases in the figures. We see from the figures that using two OSNs is sufficient to obtain close-to-perfect ROC performance for this network setting.

Finally, in Figs. 7 and 8 we investigate the effect of the number of packets used to obtain delay measurements on the bottleneck identification performance of the proposed algorithm. Plotted in the figures are the strong and weak ROC curves obtained for the case in which each component network of G has 25 nodes (i.e., $|V_u| = |V_v| = 25$), the probability of edge creation $p = 0.5$, and the number of OSN(s) packets measured $N = 5000, 1000$, and 500. The number of OSNs employed in $|S| = 2$ in all cases. From the figures, we observe that the strong ROC performance degrades sharply as the number of packets measured drops to $N = 1000$ or below. However, the weak ROC performance degrades in a much more graceful manner as N decreases. The main reason for this observed difference in degradation in performance comes from the fact that the strong false alarm event is much stricter a requirement than its weak counterpart.

V. Conclusion

The proposed bottleneck identification algorithm based on applying a SVM on clustered packet delay statistics is shown to give very good detection performance in a NoN that is modeled by a single-bottleneck graphs, when the component networks of the NoN are not too small in size or the connectivity between nodes within the component networks is not too sparse. This bottleneck identification algorithm has the advantage that it requires almost no information about the topology or traffic pattern of the underlying NoN. It only employs round-trip delay measurements of packets sent by nodes to other nodes in the NoN to perform bottleneck identification. Extensions to more complicated NoNs that contain more and unknown numbers of bottleneck links are currently under investigation.

References

[1] J. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Robustness of a network of networks," *Phys. Rev. Lett.*, vol. 107, no. 19, pp. 195 701–1 – 195 710–5, 2011.

[2] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, 1970.

[3] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser, "Min-cut clustering," *Mathematical Programming*, vol. 62, no. 1-3, pp. 133–151, 1993.

[4] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[5] G. Karypis and V. Kumar, "Multilevel k-way partitioning scheme for irregular graphs," *Journal of Parallel and Distributed computing*, vol. 48, no. 1, pp. 96–129, 1998.

[6] M. X. Cheng, Y. Ling, and B. M. Sadler, "Network connectivity assessment and improvement through relay node deployment," *Theoretical Computer Science*, vol. 660, pp. 86–101, 2017.

[7] J. Cheeger, "A lower bound for the smallest eigenvalue of the laplacian," in *Problems in Analysis*. Princeton U. Press, 1969, pp. 195–199.

[8] M. Coates, A. O. Hero III, R. Nowak, and Bin Yu, "Internet tomography," *IEEE Signal Proc. Mag.*, vol. 19, no. 3, pp. 47–65, May 2002.

[9] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[10] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-source internet tomography," *IEEE J. Select. Areas Commun.*, vol. 24, no. 12, pp. 2221–2234, Dec. 2006.

[11] C.-C. Chang and C.-J. Lin, "A library for support vector machines," *LIBSVM*, 2001.