Noninvasive Discovery of Network Traffic Patterns

Clara Cousins
ccousins@college.harvard.edu

William Drew
wdrew@college.harvard.edu

***Abstract*—**
***Index Terms*—internet, topology, flow, network, noninvasive**

I. Introduction

With an increasing reliance on internet networks to deliver content to consumers, there is a growing need for network analysis tools that enable businesses to understand their own network traffic footprint and that of their competitors. However, due to the sensitivity of network traffic information including port number, protocol type, and packet routing path, there is also a challenging demand to recover network traffic flows from information that is not embedded in the packets themselves. A noninvasive network analysis method would be revolutionary to the field. In this work, we explore how directed acyclic graphs (DAGs) can represent network traffic flows across hosts using only packet count statistics at routers.

II. Problem to Solve

This project asks: To what extent does DAG structure learning reveal computer network traffic flows based on router packet counts?

III. Background, Motivations, and Prior Work

Content delivery services, such as Netflix and Hulu, compete for consumers but may not have access to read the packets at each others routers or servers. Netflix and Hulu would be motivated to compare their traffic flow patterns among a collection of routers (such as for a certain internet service provider in the metro Boston area) in order to identify the routers that are bottlenecks for their network traffic as well as the geographical areas or parts of the network that each company reaches effectively [1].

Many techniques are currently used to identify traffic at various degrees of granularity, including the packet, individual host, or aggregate host (i.e., local area network [LAN]) levels. At the packet level, packet header information may be required to determine the source and destination addresses and ports. One such example of network traffic monitoring at the packet level is implemented in Cisco NetFlow [2]. Cisco NetFlow is an arguably invasive approach to analyzing network traffic as it requires packet header inspection using NetFlow-enabled devices. At the individual host level, network traffic can be understood for a given router by probing networks with dummy packets [3], but this method does not scale well to collections of hosts since they increase traffic and may resemble distributed denial of service attacks [4]. Across hosts, graphical models have been used to summarize traffic flows. Traffic dispersion graphs (TDGs), used by Iliofotou et al (2007), treat IP addresses as nodes and use directed edges to represent node-interactions (i.e., traffic among hosts) over a given time interval. TDGs for HTTP traffic may be derived by inspecting the port number of certain packets, such as TCP SYN packets, and setting an edge between the source and destination hosts for that packet if port 80 is used at the destination [5]. The topology of the TDG provides a useful representation of traffic aggregated at the host level.

We aim to solve the problem of how to represent network traffic in a local area network (LAN) at the level of aggregate host flows without using invasive route traces involving the specific servers of interest. Solving this problem would be useful for competitors seeking to understand traffic flow patterns in a LAN in order to detect bottlenecks and regions of particularly high usage even if the available data is simply packet count at routers in the LAN. This could enable content service providers to craft new subscriber services, optimize shared resource utilization, and understand the limitations of the content delivery network.

IV. Project Goals & Evaluation Metric

The goal of this project is to determine with DAG structure learning algorithms can reveal accurate traffic flow topologies using simulated networks adhering to a link state routing scheme using Dijkstras shortest path algorithm. This goal would be successfully achieved if

$$\frac{\text{True DAG edges}}{\text{Total edges in ground truth topology}} > \frac{\text{False DAG edges}}{\text{Total edges in DAG}}$$

V. Proposed Approach, Novelty and Secret Weapon

We propose a novel method to learn aggregate flows across hosts using only packet counts at routers. Critically, this approach noninvasively learns network traffic flows without examining sensitive fields such as destination IP addresses, port numbers, protocol types, and other information that can be collected from data packet headers. We accomplish this using DAG structure learning algorithms to predict network traffic flows.

VI. Intellectual Points

Directed acyclic graphs (DAGs) encode the joint probability distribution of a collection of random variables (nodes) without allowing conditional dependencies (edges), interpreted as directed causal effects, to form cycles among the nodes. Conditional independence exists between random variables

X and Y given Z with nonzero probability $(X \perp\!\!\!\perp Y|Z)$ if $P(X \perp\!\!\!\perp Y|Z) = P(X|Z) * P(Y|Z)$ where $P(A|B) = P(A,B)/P(B)$. Because DAGs are probabilistic graphical models, they provide statistical power to detect the true conditional dependencies but ultimately rely on certain assumptions to allow causality to be inferred. Three major assumptions of DAGs include the causal Markov condition, faithfulness, and causal sufficiency [6]. Briefly, the causal Markov condition, which allows edges to be interpreted as causal effects, states that a node is conditionally independent of all other nodes except for its children (direct descendants) given its parents (direct causes of the node) [7]. Faithfulness means that the DAG learned from the data contains conditional dependencies that are indeed reflected in the data and are not the result of model parameters cancelling each other during the learning process [8]. Finally, causal sufficiency ensures that enough nodes are included in the DAG such that the conditional dependencies learned have the potential to reflect real causal effects without hidden nodes being responsible for the relations [8].

In order to model network traffic flows on the aggregate host level because even if cycles exist, we are not interested in learning them because we primarily want to find sources versus destinations so want to constrain flows not to be starting and ending at the same host in a cyclic flow. Note that even though we expect most flows not to contain cycles, we still test to see if a DAG models real traffic flows in real networking data since we could a dataset that does provide routes so we can check which links are recapitulated in the learned DAG.

There are three main classes of algorithms used to learn the structure of a DAG from observational data (i.e., without manipulation of the values at any node). Briefly, score-based approaches (such as greedy hill-climbing) begin with a random DAG structure and scores it on goodness-of-fit relative to the data, then makes every possible perturbation (either adding, removing, or reversing direction of an arc) to the DAG, rescores the fit, then updates the DAG (making the perturbation that will maximize the score from the given space in the search) structure in order to maximize the score, stopping once perturbations no longer exist that improve the score [9]. Various scores may be used in hill-climbing, including the log-likelihood, Bayesian Information Criterion (BIC), and Akaikes Information Criterion (AIC), with the latter two being regularized versions of log-likelihood.

Constraint-based algorithms, such as the PC algorithm [10], identify directed edges in the DAG by conditional independence testing. Specifically, constraint-based algorithms begin with saturated DAGs (having every possible causal relation indicated) and eliminate arcs according to the conditional independence relations in the data [9]. Different conditional independence tests exist, including Pearsons linear correlation test with Students t.

Hybrid algorithms use conditional independence testing to build initial DAGs from which score-based approaches can search from this restricted space to determine a DAG that fits the data well [9]. One hybrid algorithm is the Max-Min Hill-Climbing algorithm, which constructs the DAG skeleton by the PC algorithm and then uses this as the initial DAG from which the hill-climbing algorithm can be used to search a reduced search space to direct the edges.

The algorithms have different accuracies for different topologies, but the conditions for which each algorithm performs best are not well understood [11]. Which algorithm may learn computer network traffic flow patterns best for different topologies is not known. In order to learn aggregate flows across hosts and therefore provide insight to competitors about how traffic goes in their network, we need to identify the most appropriate algorithm and then apply them to simulated network traffic data to learn DAGs, which we can evaluate for accuracy given the known flow topology in the simulated data.

VII. Work Performed

To create simulations of network topologies, we used a python model using the NetworkX library, a library commonly used to study graphs and networks. We created two different 20-node networks with Random Internet Autonomous System and random tree topologies. We selected these topologies since they could conceivably be models of actual Local Area Network (LAN) topologies. In each of these networks, we defined one output node as linked to Business A servers and one other output nodes as linked to Business B servers. The remaining 18 source nodes were all possible sources of internet traffic directed toward the output nodes. The links in the network topology were given random delays between 1ms and 15ms.

To simulate network traffic between consumer nodes and servers operated by either Business A or Business B, we randomly selected pairs consisting of a source nodes and output nodes linked to Business A and Business B servers. Each simulated day had a randomly selected bias for source node selection to increase the variance in day-to-day data to help facilitate DAG learning. Traffic flow was simulated by accumulating packet counts at nodes defined within the shortest path between source and destination as determined by Dijkstras Algorithm. One hundred pings were run on each topology for each simulated day. Two hundred days of data for each topology were collected in total.

We chose a variety of algorithms that are expected to be suited to different topologies. The algorithms we used are: hill-climbing with the BIC, AIC, and log-likelihood score (each with 0 restarts and 1 perturbation per iteration), the PC algorithm with the linear correlation test (alpha level = 0.05), the fast iamb algorithm with the linear correlation test (alpha level = 0.05), and the max-min hill-climbing with the linear correlation test (alpha level = 0.05) and BIC score (0 restarts and 1 perturbation per iteration).

We then attempted to learn the network topology with DAG learning algorithms. Algorithm performance was evaluated through a combination of two metrics; TFT is the proportion of true found edges to total true edges, FFF is the proportion of falsely found edges to total found edges. A well-performing algorithm should have a high TFT indicating that the algorithm

discovered many edges in the original topology and a low FFF indicating that the algorithm found as few extraneous edges as possible.

VIII. Results and Discussion

IX. Conclusion

References

[1] **TODO**
[2] Cisco. “Introduction to the Cisco TOP NetFlow”. May 2012.
[3] **TODO**
[4] Donnet, B., & Longstaff, I. (2007). “Combining MIMO Radar with OFDM Communications”. 2006 European Radar Conference, 37-40.
[5] Iliofotou, M., Pappu, P., Faloutsos, M., Mitzenmacher, M., Singh, S., & Varghese, G. (2007). “Network monitoring using traffic dispersion graphs (tdgs)”. Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, 315-320.
[6] Markus Kalisch, Martin Machler, Diego Colombo, Marloes H. Maathuis, & Peter Buhlmann. (2012). “Causal Inference Using Graphical Models with the R Package pcalg”. Journal of Statistical Software, 47(11), 1-26.
[7] Hausman, D., & Woodward, J. (1999). “Independence, invariance and the causal Markov condition”. The British Journal for the Philosophy of Science, 50(4), 521-583.
[8] White, A., & Vignes, M. (2018). “Causal Queries from Observational Data in Biological Systems via Bayesian Networks: An Empirical Study in Small Networks”. ArXiv.org, ArXiv.org, May 4, 2018.
[9] Nandy, P., Hauser, A., & Maathuis, M. (2018). High-dimensional consistency in score-based and hybrid structure learning. Annals of Statistics, 46(6A), 3151-3183.
[10] Spirtes, P., Glymour, C., & Scheines, R. (2000). Causation, prediction, and search / Peter Spirtes, Clark Glymour, and Richard Scheines ; with additional material by David Heckerman ... [et al.] (2nd ed., Adaptive computation and machine learning). Cambridge, Mass.: MIT Press.
[11] **TODO**
[12]
[13]