LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-421451

Link Homophily in Application Layer and its Usage in Traffic Classification

B. Gallagher, M. Iliofotou, T. Eliassi-Rad, M. Faloutsos

December 15, 2009

2010 IEEE INFOCOM Conference
San Diego, CA, United States
March 15, 2009 through March 19, 2009

Disclaimer

Link Homophily in the Application Layer and its Usage in Traffic Classification

Brian Gallagher* Marios Iliofotou† Tina Eliassi-Rad* Michalis Faloutsos†
*Lawrence Livermore National Laboratory
{bgallagher, eliassi}@llnl.gov
†University of California Riverside
{marios, michalis}@cs.ucr.edu

***Abstract*—This paper addresses the following questions. Is there *link homophily* in the application layer traffic? If so, can it be used to accurately classify traffic in network trace data without relying on payloads or properties at the flow level? Our research shows that the answers to both of these questions are affirmative in real network trace data. Specifically, we define link homophily to be the tendency for flows with common IP hosts to have the same application (P2P, Web, etc.) compared to randomly selected flows. The presence of link homophily in trace data provides us with statistical dependencies between flows that share common IP hosts. We utilize these dependencies to classify application layer traffic without relying on payloads or properties at the flow level. In particular, we introduce a new statistical relational learning algorithm, called *Neighboring Link Classifier with Relaxation Labeling* (*NLC+RL*). Our algorithm has no training phase and does not require features to be constructed. All that it needs to start the classification process is traffic information on a small portion of the initial flows, which we refer to as *seeds*. In all our traces, NLC+RL achieves above 90% accuracy with less than 5% seed size; it is robust to errors in the seeds and various seed-selection biases; and it is able to accurately classify challenging traffic such as P2P with over 90% Precision and Recall.**

I. Introduction

Homophily, a concept from social sciences, asserts that similar entities tend to be related to one another. This work investigates the existence of homophily in application-layer traffic and its use in traffic classification. Specifically, we define *link homophily* to be the tendency for flows with common IP hosts to have the same application compared to randomly selected flows; and measure it on a network-wide *trace graph*. Given a network trace, we create a graph by representing individual IP hosts as nodes and communication flows between hosts as links. The application of a particular flow (e.g., P2P, SSH, Web, etc.) is then represented as a label on that link in the graph. Figure 1 depicts a pictorial representation of a partially labeled trace graph. Given such a graph, we measure link homophily by iterating over the set of links with known labels and computing the proportion of neighboring links that have the same application. In our experiments, we observe that link homophily exists in a variety of real network traces. Armed with this knowledge, we utilize the statistical dependencies between flows that share common IP hosts to classify application layer traffic without relying on payloads or properties at the flow level. This **relational** view of traffic classification treats the problem as information dissemination over the network-wide trace graph.

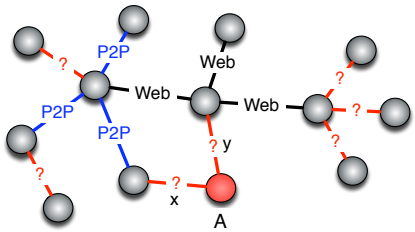


Fig. 1: Relational view of application classification. Nodes in the trace graph represent IP hosts and links represent network flows between hosts. The application class of some flows are initially known (these are the seeds), while others are unknown. Our goal is to use the initial seeds in order to infer the labels of all unknown links in the graph.

We propose a relational classifier, called *Neighboring Link Classifier with Relaxation Labeling* (*NLC+RL*), that takes as input a partially labeled trace graph (see Figure 1) and accurately identifies applications for all the unknown flows. We can obtain partial labels (a.k.a. *seed information*) in a variety of ways. Section V details different ways of obtaining seed information. NLC+RL is robust to small quantities of seeds and to errors and biases in seed labels (see Section IV-B).

The main contributions of our work are as follows. **(1)** We define link homophily on network-wide trace graphs, where (on average) links with a common endpoint tend to have the same application classes compared to randomly selected links; and show that real trace graphs exhibit link homophily. **(2)** We propose a new algorithm, NLC+RL, for traffic classification, based on techniques from the field of statistical relational learning. **(3)** We demonstrate the effectiveness of NLC+RL on both backbone and access-link traces. Our method achieves over 90% accuracy with fewer than 5% of flows initially labeled; and can classify P2P traffic (a challenging task) with over 90% Precision and Recall. **(4)** We show that our method is robust to: (a) errors introduced by the initial seed flows, achieving 80% accuracy with a seed error rate of 50% and (b) seeding biases at the host- and application-level.

Our work in perspective. By posing the traffic classification problem as a relational learning task on a trace graph, we open the door for powerful tools from statistical relational learning and graph mining, to be used for this problem. Exploiting relational dependencies among IP hosts (such as link homophily) enables us to overcome traffic obfuscation and not rely on payloads or properties at flow level.

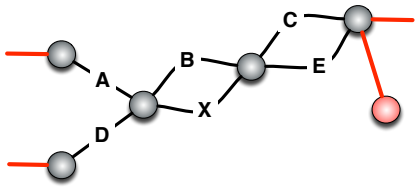


Fig. 2: Neighboring links of X are A, B, C, D, and E because they share a common endpoint.

II. Link Homophily in Application Layer Traffic

Before defining link homophily, we need to define the term *neighboring links*. We consider two links (i.e. flows) to be neighbors if and only if they share a common node (i.e. IP host). Figure 2 provides a pictorial view of neighboring links.

A. What is link homophily?

The term *homophily* (love of the same) was coined in the 1950s by sociologists. In the context of application layer traffic, we define **link homophily** as the tendency for *neighboring links* to have (on average) the same labels compared to randomly selected links. In other words, link homophily states the following:

$$P(label(l_1) \equiv label(l_2) \mid neighboring\_links(l_1, l_2)) > P(label(l_1) \equiv label(l_2) \mid random\_selection(l_1, l_2))$$

We measure link homophily by iterating over the set of links with known applications in a trace graph and computing what proportion of neighboring links have the same application. Table I reports the pseudo-code for computing link homophily per application on a graph.

LinkHomophily(G):
  /* initialize $homopilyPerClass$ array */
  for each application class $c \in C$ **do**
    $homophilyPerClass[c] = 0$;
  end for
  /* compute link homophily */
  $L = G.labeledLinks$;
  for each labeled link $l \in L$ **do**
    $N = l.neighboringLinks$;
    $homophily[l] = \frac{count(\forall n \in N : label(l) \equiv label(n))}{|N|}$;
    $homophilyPerClass[label(l)] += homopily[l]$;
  end for
  /* normalize link homophily per application class */
  for each application class $c \in C$ **do**
    $homophilyPerClass[c] = \frac{homophilyPerClass[c]}{count(\forall l \in L : label(l) \equiv c)}$;
  end for

TABLE I: Pseudo-code for computing link homophily on a trace graph, G.

B. Do trace graphs exhibit link homophily with respect to application layer traffic?

To check whether real trace graphs exhibit link homophily, we examined two internet backbone traces and two access-link traces. The backbone traces are: (1) from a Tier-1 ISP link (PAIX) and (2) from a transpacific link (WIDE). The access-link traces are: (1) from the border router of an enterprise network (ENTP) and (2) from a University in Korea (KEIO). These traces represent a diverse set of network environments, collected at different geographic locations and times. Section IV-A1 describes these traces in details.

Application *App*	PAIX	WIDE	ENTP	KEIO
P2P	1	1	0.83	0.87
Web	0.98	0.98	0.83	0.91
DNS	0.97	0.97	0.97	0.96
Chat	0.91	0.91	0.40	0.59
Mail	0.86	0.86	0.35	0.60
SNMP	0.85	0.85	0.76	0.89
FTP	0.75	0.72	0.52	0.79
SSH	0.21	0.26	0.35	0.66

TABLE II: Link homophily per application type: Probability of a random *App* link having a neighboring link of type *App*

Application *App*	PAIX	WIDE	ENTP	KEIO
P2P	0.31	0.01	0.02	0
Web	0.25	0.05	0.33	0.20
DNS	0.16	0.33	0.22	0.47
Chat	0.02	0	0.01	0
Mail	0.03	0.03	0	0.08
SNMP	0	0	0.04	0
FTP	0	0	0	0
SSH	0	0	0	0

TABLE III: Prior probability per application type: Probability of a random link of any type having a neighboring link of type *App*

Tables II and III, respectively, report link homophily per application type and prior probability per application type for all four traces. In all traces (whether backbone or access-link) and for all eight of our application types, link homophily is much higher than prior probability. For example, in the PAIX backbone trace the probability of a randomly selected Chat link having a neighboring link of type Chat is 0.91, while the probability of a randomly selected link of any traffic type having a neighboring link of type Chat is 0.02.

C. What are the origins of link homophily in application layer traffic?

We discuss the origins of link homophily by dividing the applications into two significant types: (a) client-server and (b) collaborative. In client-server applications, we expect to see "stars" in the graph: a server surrounded by clients. Clearly, these flows share a node and are of the same application, thus contributing to link homophily. In collaborative applications (such as P2P), nodes connect with multiple collaborators since the power of these applications rely on rich connectivity. This behavior also supports the observed link homophily. In addition to P2P, the same argument holds for distributed communities, some online games, and semi-structured and hierarchical applications such as DNS. We observed link homophily even in hosts with many different applications. Moreover, link homophily is often asymmetric, where one end-

point exhibits higher link homophily than the other endpoint.

III. Using Link Homophily in Traffic Classification

The presence of link homophily in trace data provides us with statistical dependencies between flows that share common IP hosts. We propose a new statistical relational learning algorithm, called **Neighboring Link Classifier with Relaxation Labeling** (**NLC+RL**), which utilizes these dependencies to classify application layer traffic without relying on payloads or properties at the flow level. Specifically, NLC+RL takes as input a *partially labeled trace graph* and infers labels for the unlabeled links by exploiting link homophily. NLC+RL is an adaption of the simplest and fastest available node-based relational classifiers [1] for the task of link classification.

The **Neighboring Link Classifier** (**NLC**) in NLC+RL assigns a label to each unlabeled link, u, based on the class frequencies observed in the set of u's neighboring links. To prevent unduly favoring nodes with many links, NLC calculates the neighboring class frequency for each of u's endpoint nodes separately and then averages the two.

When an unlabeled link u has no neighboring links that are labeled, NLC will end up with no label for u (because $P(label(u) \equiv c|u) = 0$ for all applications c). In such cases, we use the prior probability distribution observed in the initial set of labeled links to assign application probabilities to u – i.e., $P(label(u) \equiv c|u) = P_{prior}(c|$ initial set of labeled links) for all applications c. We then select the label with the highest probability.

Table IV outlines the pseudo-code for NLC. For each unlabeled link, the output of NLC is a probability distribution over the application classes – i.e., $\forall u \in U$ & $\forall c \in C$: $P(label(u) \equiv c|u)$, where U is the set of unlabeled links in the trace graph and C is the set of application classes that we want to classify (e.g., C = {P2P, DNS, Web, Chat, SNMP, FTP, SSH, Mail}). To obtain a classification for an unlabeled link, we select the application with the highest probability on that unlabeled link: $\forall u \in U : label(u) = argmax_{c \in C}(P(label(u) \equiv c|u))$.

NLC+RL is essentially a systematic method to repeat NLC multiple times in order to improve classification performance when seed information is scarce. In particular, NLC+RL augments NLC with *linked-based collective classification* by using the *relaxation labeling* (RL) algorithm [2].Linked-based collective classification refers to the combined classification (i.e., simultaneous inference) of a set of neighboring links. Two types of information are utilized in linked-based collective classification: (1) correlations between the label of link l and the known labels of its neighboring links, and (2) correlations between the label of link l and the unknown labels of its neighboring links.

For each unlabeled link in the partially labeled trace graph, RL maintains a current estimate of the probability distribution over the set of application classes C that we are interested in. Initial probability estimates are assigned as follows. For each labeled link, RL assigns a probability of 1.0 for the link's (application) label and a probability of 0.0 for all other (application) labels. For each *unique*unlabeled link, RL assigns the prior probability distribution observed in the initial set of labeled links. Then, each unlabeled link's probability distribution is updated t times. On each iteration, NLC is used to update the probability distributions of links, based on the current assignments of their neighboring links. In other words, RL stores probability estimates at iteration t and updates estimates for all links at iteration $t+1$. Since each link in the graph has an associated probability distribution over the set of

NLC(G):
  $L = G.labeledLinks$; /* $\mid L \mid > 0$ */
  $U = G.unlabeledLinks$;
  for each unlabeled link $u \in U$ **do**
    $N = u.labeledNeighboringLinks$;
    $N_s = u.labeledNeighboringLinksFromSrc$;
    $N_d = u.labeledNeighboringLinksFromDst$;
    if ($\mid N' \mid > 0$) **then**
      $p_s = \frac{count(\forall n \in N_s \& \forall c \in C: label(n) \equiv c)}{|N_s|}$;
      $p_d = \frac{count(\forall n \in N_d \& \forall c \in C: label(n) \equiv c)}{|N_d|}$;
      $P(c|u) = \frac{1}{2}(p_s + p_d)$;
    else
      $P(c|u) = \frac{count(\forall l \in L \& \forall c \in C: label(l) \equiv c)}{|L|}$;
    end if
  end for

TABLE IV: Pseudo-code for Neighboring Link Classifier (NLC). Here $ep(u)$ is the set of endpoint nodes of link u.

NLC+RL(G):
  $L = G.labeledLinks$;
  $U = G.uniqueUnlabeledLinks$;

  /* initialize probability estimate for labeled links */
  for each labeled link $l \in L$ **do**
    $P_0(c|l) = 1$ **if** $label(l) \equiv c$;
    $P_0(c|l) = 0$ **otherwise**;
  end for

  /* initialize probability estimate for unlabeled links */
  for each unlabeled link $u \in U$ **do**
    $P_0(c|u) = \frac{count(\forall l \in L \& \forall c \in C: label(l) \equiv c)}{|L|}$;
  end for

  /* update probability distributions of unlabeled links */
  repeat
    for each unlabeled link $u \in U$ **do**
      $\forall c \in C$:
        $P_{t+1}(c|u) =$
          $\beta_{t+1} \cdot \mathbf{NLC}_{\langle t,u,c \rangle}(G) + (1 - \beta_{t+1}) \cdot P_t(c|u)$
    end for
  until $(t \equiv 99)$

TABLE V: Pseudo-code for Neighboring Link Classifier with Relaxation Labeling (NLC+RL). $NLC_{\langle t,u,c \rangle}(G)$ outputs the probability $P(label(u) \equiv c|u)$ computed by NLC after iteration t's updates on the graph G. The simulated annealing parameters are: $\beta_0 \in [0, 1]$; $\beta_{t+1} = \beta_t \cdot \alpha$; and α is a decay constant. For our experiments, we used the standard values of $\alpha = 0.99$ and $\beta_0 = 1$; and found $t = 99$ iterations sufficient for convergence.

applications C (instead of a hard label assignment), NLC will sum the probabilities of each application for each neighboring link instead of simply counting labels of each application. To catalyze convergence, we perform simulated annealing [1]. Table V outlines the pseudo-code for NLC+RL. Like NLC, we obtain a classification for an unlabeled link by selecting the application with the highest probability on that unlabeled link: $\forall u \in U : label(u) = argmax_{c \in C}(P(label(u) \equiv c|u))$.

IV. Experiments on Traffic Classification

A. Experimental Design

1) Data Sets: We evaluate NLC+RL on four real-world traces. Two traces are collected at the internet backbone, from a Tier-1 ISP link (PAIX) and from a transpacific link (WIDE). The access-link traces are collected at the border router of an enterprise network (ENTP) and from a University in Korea (KEIO). These traces represent a diverse set of network environments (backbone and access-link), collected at different geographic locations and points in time.

Table VI lists the distribution of flow-types for each of the four traces. We define a flow using the well-known 5-tuple ⟨`srcIP`, `srcPort`, `dstIP`, `dstPort`,`protocol`⟩. Bidirectional flows are represented as undirected links in the trace graph and are reported as single flows in Table VI. All our traces contain payload information, thereby enabling us to label the flows using signature-matching techniques described in [3] and later enhanced in [4]. For each traces, we classify traffic into approximately 15 traffic categories. In Table VI, we report detailed statistics for the following eight main classes: DNS, Chat, FTP, Mail, P2P, SNMP, SSH, and Web. These 8 classes represent the majority of the known traffic as we show in Table VI. The remaining classes (reported as *Rest* in Table VI) include network games and other applications that contribute less to the overall traffic. In our evaluation, we include all classes in the trace graph. From our analysis we removed all flows that did not carry any payload. Such flows represent worm scanning activity and other failed TCP connections [3], [4].

a) Backbone Traces: **PAIX:** This data set was collected from an OC48 link of a commercial US Tier-1 ISP at the Palo Alto Internet eXchange (PAIX), connecting San-Jose with Seattle. For the experiments shown here, we used a temporal 30-second sample collected from 18:00 to 18:00'30 during April 21st 2004. We also tested our methods with other 30-second samples, which produced similar classification results. More details regarding the sample duration are provided in Section V. The monitor captured traffic from both directions of the traffic link. In addition, the data set contains up to 16 bytes of payload from each packet. **WIDE:** This trace is collected from a transpacific backbone link connecting the US with Japan and carries commodity traffic of the WIDE member organization. For the experiments show here we used a temporal 5-minute sample collected from 22:45 to 22:50 on the March 3rd 2006. The trace contains traffic from both directions of the link. For each packet, it contains full packet header and 40 bytes of payload.

	Backbone Traces		**Access Link Traces**	
	PAIX	**WIDE**	**ENTP**	**KEIO**
Application Traffic Mix				
P2P	76055 (31%)	893 (1%)	3780 (2%)	79 (0%)
Web	62860 (25%)	5877 (5%)	88883 (33%)	6868 (20%)
DNS	39387 (16%)	39532 (33%)	58158 (22%)	16498 (47%)
Chat	4794 (2%)	734 (0%)	1953 (1%)	140 (0%)
Mail	6987 (3%)	2973 (3%)	1307 (0%)	2958 (8%)
SNMP	94 (0%)	9 (0%)	10485 (4%)	52 (0%)
FTP	152 (0%)	46 (0%)	864 (0%)	2 (0%)
SSH	18 (0%)	2 (0%)	509 (0%)	7 (0%)
Rest	9215 (4%)	543 (0%)	1942 (1%)	352 (1%)
Unknown	47442 (19%)	68946 (58%)	99954 (37%)	8372 (24%)
Trace Graph Information				
Year	2004	2006	2007	2006
#Nodes	171641	101264	57285	24994
#Links	247004	119553	266878	35328
% in LCC	87%	90%	99%	91%
Duration	30 seconds	5 minutes	1 hour	5 minutes

TABLE VI: Summary of our backbone and access-link traces.

b) Access-Link Traces: **ENTP:** This data was collected at the perimeter of an access-link network over a five-day period in 2007. For our experiments, we use a temporal sample of the data, collected between 10 AM and 11 AM on the third day of the capture. This hour was one of the busiest during the five-day period. **KEIO:** This trace was collected from a link inside Keio University Shonan-Fujisawa campus during 2006. For our experiments, we use a temporal 5-minute sample of the data, collected on August 10th from 1:20AM to 1:25AM. This trace also contains detailed packet header information as well as the first 40 bytes from each packet.

For all four of our traces (PAIX, WIDE, ENTP, and KEIO), we tested NLC+RL with other temporal samples. The outcome in terms of classification performance were similar.

2) Trace Graphs: For each data set, we create a trace graph with nodes representing hosts (IP addresses) and links representing communications (flows) between hosts, as shown in Figure 1. Our trace graphs allow multiple links between two nodes (see Figure 2) given that each link represent a different flow (i.e., uses different port numbers or protocol).

The graph sizes are listed in Table VI as number of nodes and links. The size of the Largest Connected Component (LCC) in the graph is reported as the percentage of flows that belong to it. From Table VI, we see that there is one large connected component that contains the majority of links ($> 87\%$) in the graph. All connected components of the graph

contain a diverse mix of links from various applications.

Since NLC+RL is not applicable to communications in isolation, we remove such links from the trace graph. Isolated communication-links do not have any neighboring links during the observation interval (i.e., both of their end-hosts do not communicate with any other hosts). These isolated links are a very small portion of the entire graph ($< 1\%$ for all traces). Section V provides more details on these links.

3) Obtaining Seed Information: To start the classification process, our method requires a small amount of seed information (which is common in supervised learning approaches). In order to obtain seed labels for our experiments, we *emulate* the existence of a **seed provider**, using a payload-based signature-matching method similar to previous works [5], [3], [6]. It compares the payload of each packet to a predefined set of signatures for application-layer traffic such as P2P, DNS, Games, Chat, Web, Mail, etc. Traffic that does not match the predefined set of signatures is labeled "unknown."

It is important to note that our approach is not tied to any particular seed provider (see Section V for details). In Sections IV-B2 and IV-B3, we present results that show the robustness of NLC+RL to biases and errors in seed labels.

4) Experimental Methodology: For all of our results, the basic experimental setup is the same: we run 10 trials and report the average performance. The details of our experimental methodology are as follows.

For evaluation purposes only, we need ground-truth on the flows in our trace data, which we obtain using the payload-based signature-matching method described in Section IV-A3. We assume this method informs us of the "true" application of a flow. However, since it is signature-based, the payload classifier is unable to classify every flow in a trace graph. Flows with unknown applications are still included in the trace graph, allowing them to propagate traffic information, and NLC+RL will provide labels for all flows in a network. However, we can only evaluate the performance of NLC+RL on the known portion of the traffic.

To test NLC+RL, we vary the number of labeled links (i.e., the initial seed). Specifically, we vary the proportion of links that have seed labels from 1% to 90% of the total number of links in the trace graph. Only these links retain their labels. All remaining links have their labels stripped and are used to evaluate the classifiers' performances. We refer to the proportion of links that have seed labels as **Seed Size, s**.

For each seed size, we run 10 trials and report the average performance. For each trial and seed size, we choose a class-stratified random sample containing $s\%$ of the total links in the graph. These links retain their labels. All other labels are removed. We then evaluate on all unlabeled links for which ground truth is available. To be fair across classifiers, we use identical labeled- and unlabeled-link splits for each classifier. We evaluate classifier performance using the standard metrics of Accuracy (ACC), Precision (P), Recall (R), and F1-score (the harmonic mean of Precision and Recall).

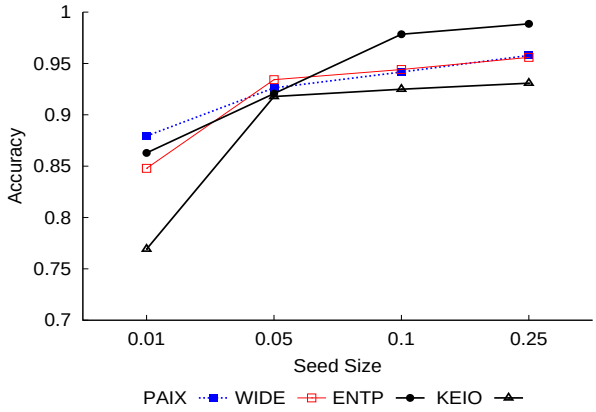


Fig. 3: Accuracy of NLC+RL over all 4 traces.

B. Experimental Results

Here, we present experiments that answer these questions:

- **Q1:** Can NLC+RL perform well even with limited seed information? **Answer:** Yes, even with only 5% of links labeled, NLC+RL achieves over 90% accuracy.
- **Q2:** How is the per-class performance of NLC+RL? **Answer:** NLC+RL performs very well over a large range of application classes. Even for challenging applications, such as P2P, it can achieve over 95% F1-score with 20% initial seed over all four traces.
- **Q3:** How sensitive is NLC+RL to errors made by the seed provider? **Answer:** Even with 50% erroneous seed links, NLC+RL can still achieve 80% classification accuracy over the remaining links.
- **Q4:** How sensitive is NLC+RL to biases in the selection of initial seed when: (a) we know nothing for some hosts and everything about others (host-biased seeding), and (b) we know all the flows for some applications, e.g., DNS, but have only a limited knowledge for others, e.g., P2P (application-biased seeding)? **Answer:** NLC+RL's performance is robust to both types of seeding biases.
- **Q5:** Can NLC+RL accurately classify traffic of hosts with multiple applications? **Answer:** Yes. NLC+RL represents associations at the flow-level and not at the host-level. This allows different flows of a single host to have different neighborhoods and therefore be associated with different applications.

We report results on NLC+RL and a Default classifier. The Default method classifies flows by only using the prior probability distribution on traffic mix obtained from the seed labels. For example, by looking at Table VI, we observe that the prior probability of a traffic being P2P in PAIX is 31%. Intuitively, the Default method shows how easy or hard is the classification task at hand. For example, if 99% of the flows are Web, by just predicting Web for all flows we achieve 0.99 accuracy. We also report comparative results with BLINC, CoralReef, and a SVM flow-based classifier.

1) Classification Performance with Limited Seed Data and per Application: Figure 3 depicts the accuracy of NLC+RL for

all four traces. **NLC+RL performs very well given modest seed sizes** – e.g., its accuracy is above 90% when seed size is greater than or equal to 5%.

NLC+RL performs well across the range of application classes examined in our data sets. This is particularly important for applications such as P2P, which can be more challenging [7], especially at the backbone [4]. Figure 4 summarizes our results. For brevity we will report only one example of each category (backbone and access-link) given that the other results are qualitatively similar. We will refer to individual traces to highlight any differences.

For the PAIX trace, Figure 4a reports F1 scores for NLC+RL. We observe that even with a small seed size, NLC+RL has a very good classification performance for the dominant applications in our traces: P2P, Web, and DNS. The F1 score of all three dominant applications is over 0.9 even with 10% seed size. The exception is NLC+RL's performance on SSH, which can be attributed to the extremely small number of SSH flows (18 flows) in this backbone trace ($<$0.01% of the total flows).

For the ENTP trace, Figure 4b reports F1 scores for NLC+RL. It is interesting to observe that the traffic types appear to divide naturally into two sets: a **higher performance set (HP)**, made up of P2P, DNS, FTP, SNMP, and Web traffic, and a **lower performance set (LP)**, comprised of Mail, SSH, and Chat traffic. NLC+RL achieves better F1 scores across the board on the HP set than the LP set. Furthermore, as the seed size drops, the gap in performance between HP and LP grows. Note that this difference in performance is due to differences in both Precision and Recall, although the differences in Recall are larger overall. So, when NLC+RL makes mistakes, it tends to be misclassifications of Mail, SSH, and Chat traffic, which have the most class heterogeneity among their neighboring links. For these classes, we may benefit from more sophisticated algorithms, which learn statistical dependencies between neighbors instead of simply relying on link-homophily [8].

In both traces, NLC+RL gives very good results for challenging application such as P2P and games. With only 5% seed size, NLC+RL achieves above 95% Precision and Recall in identifying the 3,500 flows of the HalfLife online game present in the PAIX trace. Using 5% seed size, NLC+RL also classifies P2P flows with above 91% Precision and 98% Recall. Kim et al. [4] reported that BLINC achieved less than 5% Recall on identifying HalfLife, and 78% Precision with 83% Recall in identifying P2P on the same backbone trace.

2) Robustness to Seed Provider Errors: To test the robustness of NLC+RL to errors made in the seed labels, we run the following procedure. We take the seed labels provided by payload-based signature-matching method and alter them separately for each of the 10 trial runs. The reported seed provider's accuracy is $1.0 - p$, where p is the probability of alteration for each known label. Each link has a probability p of being altered on a given trial run. If a label is selected for alteration, its original value is removed and a new value is assigned at random, based on the frequency of occurrence of each label in the original set of seed labels. So, more frequent labels are more likely to be assigned to an altered label, but the process is stochastic. (The original label is never re-assigned to an altered label. Otherwise, this would not be an error.)

NLC+RL is robust to errors in seed information, even when those errors are up to 50%. Figure 5a shows the performance of NLC and NLC+RL on the PAIX trace when there are errors in the underlying seed provider. Results for the ENTP trace are shown in Figure 5b. We observe that both relational classifiers are quite robust to seed provider's errors. Also, the accuracy of both classifiers falls off more slowly than the accuracy of the seed provider. Even when the seed provider makes an incorrect classification as often as it makes a correct classification (i.e., 50% accuracy), NLC+RL achieves 80% accuracy over all four traces. So, NLC+RL is able to make good use of the imperfect information that the seed provider is providing. Also, recall that NLC+RL is making predictions on flows that the seed provider is unable to classify. So, NLC+RL is actually able to obtain higher accuracy than the seed provider, while classifying links that the seed provider was unable to classify.

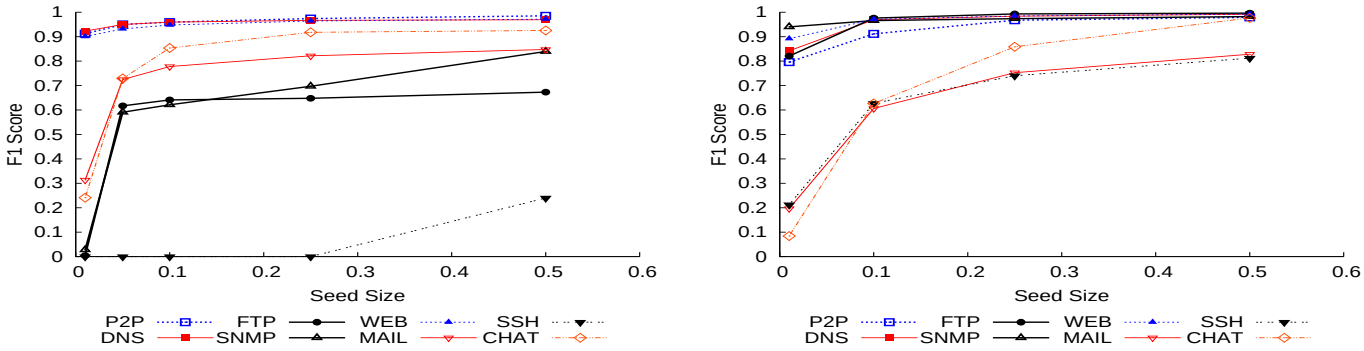
3) Robustness to Bias in Provided Seeds:

a) Host-biased Seeding: We study the behavior of NLC+RL when seed labels are withheld preferentially by host, rather than uniformly at random. This demonstrates the robustness of NLC+RL in the face of (1) intentional attempts by certain hosts to obfuscate their traffic and (2) seed provider classifiers that are prone to such biases (e.g., host-based methods such as BLINC [3]).

Our procedure for host-biased seeding is as follows. Instead of choosing links at random to unlabel, we choose a host H at random and then unlabel a specified percent P of H's flows. We repeat this process until only the specified number seed labels remain in the overall network.

In Figure 6, we show the performance of NLC+RL over all traces for $P = 100\%$. As it can be seen from the figure, NLC+RL performs extremely well on all data sets down to 50% seed size, even with hosts obfuscating 100% of their traffic. Note that this task is especially difficult since the flows we evaluate on belong to hosts that we know essentially nothing about (since they have no labeled flows to begin with). While performance does decline as more hosts begin to hide their traffic, the level of performance remains impressive down to 5% seed size for the majority of data sets. By allowing a small fraction of a host's flows to be labeled ($P = 99\%$), we achieved even better results. For example, using 5% seed size the accuracy on the ENTP trace increased from 59% (with P=100%) to 75% (with P=99%). This suggests that NLC+RL can capitalize on a small fraction of known flows for a host to improve classification performance.

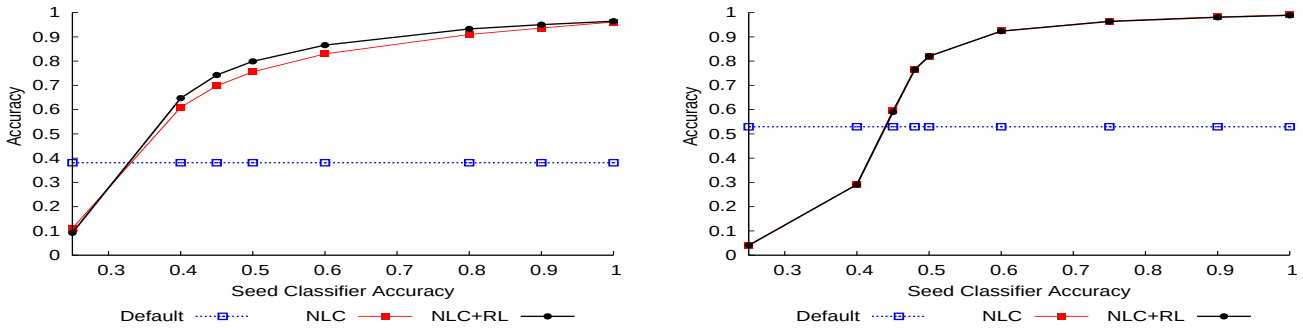
b) Application-biased Seeding: Here, we study the effects of application-biased seeding on the performance of NLC+RL. We want to simulate the real-world situation where certain classes of application traffic (e.g., P2P) are more frequently obfuscated than others. For this experiment, we assume that all flows for non-P2P applications (e.g., Web, DNS, Mail) are successfully labeled by a seed classifier, while



(a) NLC+RL on PAIX trace: F1 scores plot.

(b) NLC+RL on ENTP trace: F1 scores plot.

Fig. 4: Detailed results by class with varying seed sizes.



(a) Backbone trace (PAIX).

(b) Enterprise trace (ENTP).

Fig. 5: Accuracy of NLC and NLC+RL with 50% seed size and varying error rates in the seed information.

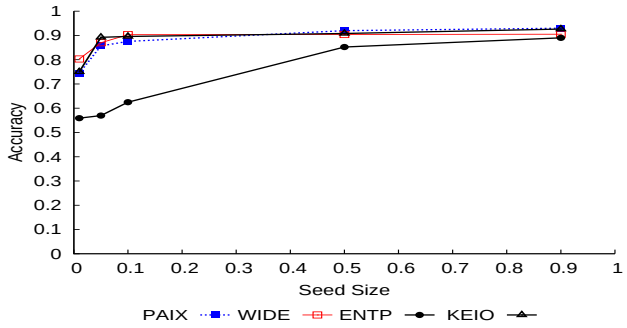


Fig. 6: The effect of host-biased selection on NLC+RL's accuracy as seed size varies.

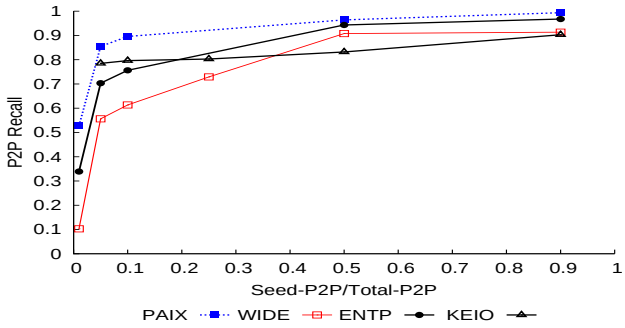


Fig. 7: The effect of P2P application-biased selection on NLC+RL's accuracy as seed size varies

the proportion of available P2P seed labels is varied. This creates a strong bias in the proportion of P2P seeds available relative to the seeds available for other classes. Note that the test set contains only P2P traffic and the training set contains very little P2P traffic. This creates an extremely difficult classification task, where the distribution of classes in the labeled training set differs widely from the class distribution in the unlabeled test set.

Figure 7 shows Recall of P2P traffic over all four traces for NLC+RL over all 4 traces. Note that Precision is fixed at 1.0 since all test instances are P2P. With less than 10% of P2P traffic known, we can correctly identify above 75% of P2P traffic in all traces, except for WIDE where Recall is approximately 60%.

We observed similar performance when selectively obfuscating other traffic types (e.g., DNS, Web, etc). We omitted those results due to space limitations. Application classes with very few flows (such as SSH in PAIX and SNMP/FTP/SSH on WIDE) have lower performance than others. This is expected since these classes represent a really small fraction of the entire traffic. Our overall results are very promising showing that NLC+RL is robust to seeding biases that can potentially be

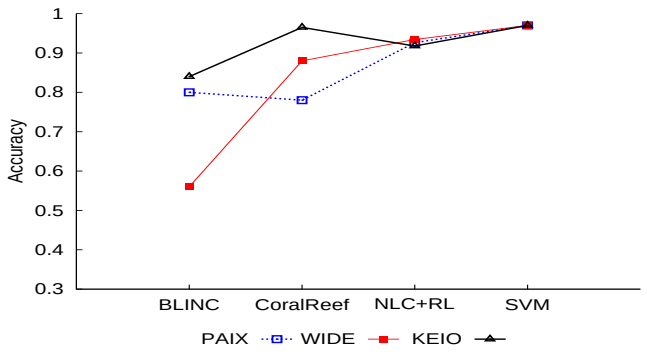


Fig. 8: Comparing NLC+RL with 5% seed size to CoralReef, BLINC, and SVM on the same traces. CoralReef and SVM rely heavily on port numbers and flow-level data (which can be obfuscated), while NLC+RL does not require such information.

observed in real-life situations.

4) Comparison with Other Methods: To establish a reference point to the performance of NLC+RL, we compare it with three other methods that operate at the: (a) port-level, CoralReef [9], (b) host-level, BLINC [3], and (c) flow-level (e.g., packet sizes), using the Support Vector Machines (SVM) learning algorithm [4]. The results from CoralReef, BLINC, and SVM are as reported by Kim et al. [4] on our common set of traces, namely: PAIX, WIDE, and KEIO. For tuning BLINC [4], they have manually adjusted the 28 parameters of BLINC so that the overall accuracy will be the highest. During their training phase, all parameters were selected after several trial-error efforts over the entire trace.

Figure 8 shows the comparison results. We ran NLC+RL with 5% initially labeled links. As we observe from Figure 8, our approach has better accuracy than BLINC over all our traces. Our performance is also better than CoralReef on all traces with KEIO being a notable exception. Note that CoralReef in this case outperforms the more sophisticated SVM algorithm as well. The authors in [4] attribute this behavior to the fact that the majority of flows in KEIO belong to legacy applications that consistently use their default port number. The accuracy of port numbers is a strong assumption which might not hold in the future even for the KEIO link. Note also that SVM capitalizes on flow features, such as packet size that can also be easily obfuscated (see Section VI). The results on SVM here uses 3% training sample size and its feature set includes port-numbers. When SVM was trained without using port-numbers, its overall accuracy dropped to 70%. Taking into account how simple our approach is compared to BLINC and SVMs, we can see the potential for NLC+RL in application classification and potentially other network monitoring tasks.

5) Classification Performance on Hosts with Multiple Applications: NLC+RL represents associations at the link-level and not at the host-level. This allows different links (flows) of a single host to have different neighborhoods and therefore be associated with different applications. An example of this behavior is illustrated in Figure 1 (host *A*).

In Table VII, we report the percentage of hosts in all four traces that appear to have flows from: (1) a single application (Sing.), or (2) from multiple applications (Mult.). These results correspond to only the test subset of the trace graphs, which is the portion of the graph that NLC+RL will try to predict labels for. Here, we used NLC+RL with 10% initial seed (and got similar results using other configurations). For comparison, we report the percentages of Sing. and Mult. hosts using the ground-truth (True) and NLC+RL's predictions for labels of the trace graph. The results show that NLC+RL can predict each individual flow while preserving the diversity in the applications used by hosts.

	PAIX		**WIDE**		**ENTP**		**KEIO**	
	%True	%Pred.	%True	%Pred.	%True	%Pred.	%True	%Pred.
Sing.	98.08	98.03	98.19	98.38	88.23	88.94	97.27	97.57
Mult.	1.92	1.97	1. 81	1.62	11.77	11.06	2.73	2.43

TABLE VII: The performance of NLC+RL on hosts with multiple applications.

V. Discussion

Selecting the time interval of observation. In our study, we divided the initial traces into smaller time intervals and created a graph for each interval. The appropriate duration for these time intervals depends on the intensity of the traffic. For example, for the ENTP trace we used a one-hour interval, which led to a graph of roughly 57K nodes and 266K edges. For the PAIX backbone trace, we experimented with multiple intervals of 30 seconds, which led to graphs of approximately 170K nodes and 250K edges. We also experimented with 10-second, 20-second, 60-second, and 5-minute intervals on the backbone trace. As expected, NLC+RL produced better performance on intervals with higher traffic intensity; these intervals are typically longer than 20 seconds for PAIX.

Seed Provider: possibilities and limitations. NLC+RL does not depend on any specific source for its initial seed information and can overcome errors in the initial seeds. In practice, the seed information can come from a plethora of possible avenues. (a) We can use *external knowledge sources* such as information from sys-admin, legacy IPs, and well-known servers. (b) We can use any *other standalone traffic classifiers* in the literature. In this case, we could even see our method as a last step for improving classification results. Also, if we have multiple classifiers, we could use them all by employing a critical synthesis. For example, in the case of classification conflicts, we could incorporate the classifier with the better accuracy or precision, especially since some methods are better for different applications. (c) We can query *search engines* for IP addresses that we want to classify, in essence, "Googling the Internet" [7], thus harnessing the power of the Web. (d) We can use *active and passive measurement techniques* for the seeding process. For example, tools could periodically join P2P applications and online games to collect IPs of potential users.

Connectivity obfuscation. Our approach is robust to packet- and flow-level obfuscation in terms of making the life

of the application more difficult in the following ways. (a) If the application is forced to *not* communicate as much as it wanted, we claim success. This reduction can curtail the annoyance, the speed of spreading, or intensity of communication. For example, it may turn out that a P2P client may have to limit its number of neighboring nodes, if it does not want to be caught. (b) If the application is forced to communicate with more IP nodes than it did before, we claim success. This increase in the communication shows up as more flows in our trace, thus becoming more visible, which could make its detection easier.

Isolated Links. In our traces, only 1% of the links were isolated (i.e. a single flow without any neighboring links). We can further reduce the number of isolated links and create larger connected components by increasing the interval of observation. Alternately, NLC+RL can classify these isolated links by using the prior probability of application classes in the graph – e.g., with 80% chance it is Web flow, given that 80% of our flows are Web flows.

VI. Related Work

Graphs have been used previously to represent network traffic for other tasks besides traffic classification. For instance, Ellis *et al.* [10] and Xie *et al.* [11] used graphs to detect the tree-like pattern of a propagating worm. Tan *et al.* [12] used graphs to group similar hosts in enterprise networks. Others [13], [14], [15] have studied the properties of IP-to-IP interaction graphs, such as the degree distribution, connected components, etc. **However, none of the previous works examined homophily, its presence in real-world trace graphs, and its utilization for traffic classification.**

Traffic classification is a well-studied problem with significant previous work. According to the level of observation, traffic classification methods can be divided into four groups. (a) Packet-level, where methods use well-known port numbers [9]. (b) Flow-level, where methods utilize flow-level features to detect applications by first specifying a set of features such as packet sizing information, packet inter-arrival times, flow duration, etc; and then using machine learning methods (either supervised or unsupervised) to automate the learning and classification processes [16], [4]. (c) Host-level, where methods first identify the role of a host (e.g., a Web server) and then label its flows accordingly. These are prone to applications (such as P2P) that have dynamic behavior in terms of randomized ports across flows [3], [17], [7]. (d) Payload-level, where methods rely heavily on using available documentation, intuition, and manual reverse-engineering of the protocols to create signatures for various applications [5], [18], [19]. **To our best knowledge, no one has viewed the traffic classification as a relational learning problem.**

VII. Conclusions

We observe homophily in application-layer traffic of real trace data, which provide us with statistical dependencies between flows that share common IP hosts. We utilize these dependencies in a relational learning algorithm, NLC+RL, to accurately classify applications of interest in network-wide trace graphs. NLC+RL is the first method to formulate the traffic classification problem as a relational learning problem. It has several attractive features: (a) *Resistance to signature, padding and timing obfuscation techniques.* A key design property is that we do not use any flow level behavior and properties in our statistical inference. This makes our approach robust to variations and obfuscation of flow level properties. (b) *Robustness to errors and biases of the initial seed information.* Although our approach relies on an initial set of classified flows, it is robust to errors in the initial seed (approximately 80% classification accuracy even with 50% error rate). In addition, NLC+RL is robust to various biases introduced by the initial selection of seeds (such as host- and application-level biases). (c) *High accuracy on application classification in real-world network traces.* Our method achieved excellent classification performance on experiments with real traces from both the backbone and access-links, achieving above 90% accuracy in all our traces with less than 5% of flows labeled.

References

[1] S. Macskassy and F. Provost, "Classification in networked data: A toolkit and a univariate case study," *MLJ*, vol. 8, pp. 935–983, 2007.

[2] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation operations," in *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, 1976, pp. 420–433.

[3] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *ACM SIGCOMM*, 2005.

[4] H.-C. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classificatoin demystified: Myths, caveats, and the best practices," in *ACM CoNEXT*, 2008.

[5] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?" in *IEEE GLOBECOM*, 2004.

[6] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *WWW*, 2004.

[7] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Unconstrained endpoint profiling (Googling the Internet)," in *ACM SIGCOMM*, 2008.

[8] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos, "Using ghost edges for classification in sparsely labeled networks," in *ACM KDD*, 2008.

[9] CAIDA Org., "The CoralReef Project, http://www.caida.org/tools/measurement/coralreef/."

[10] D. Ellis, J. Aiken, K. Attwood, and S. Tenarglia, "A behavioral approach to worm detection," in *ACM CCS WORM*, 2004.

[11] Y. Xie, V. Sekar, D. Maltz, M. Reiter, and H. Zhan, "Forensic analysis of epidemic attacks in federated networks," in *IEEE ICNP*, 2006.

[12] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek, "Role classification of hosts within enterprise networks based on connection patterns," in *USENIX Annual Technical Conference*, 2003.

[13] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs (TDGs)," in *ACM IMC*, 2007.

[14] M. Latapy and C. Magnien, "Complex network measurements: Estimating the relevance of observed properties," in *IEEE INFOCOM*, 2008.

[15] M. Meiss, F. Menczer, and A. Vespignani, "On the lack of typical behavior in the global web traffic network," in *WWW*, 2005.

[16] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[17] W. John and S. Tafvelin, "Heuristics to classify internet backbone traffic based on connection patterns," in *ICOIN*, 2008.

[18] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: Automated construction of application signatures," in *ACM MineNet*, 2005.

[19] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *ACM IMC*, 2006.

This work performed under the auspices of the U.S. DOE by LLNL under Contract DE-AC52-07NA27344.