# 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*—We address 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.

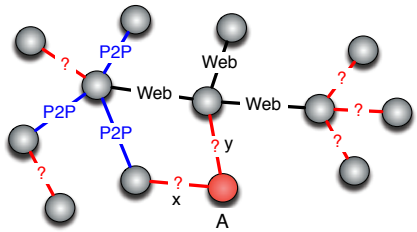


Fig. 1: Relational view of application classification. Nodes in the trace graph represent IP hosts and links represent network flows between hosts. The application class of some flows are initially known (a.k.a. the seeds), while others are unknown. Our goal is to use the initial seeds to infer the labels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

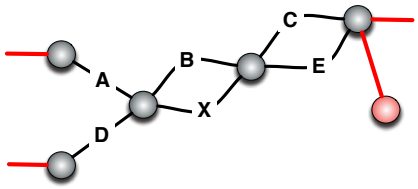


Fig. 2: Neighboring links of X are A, B, C, D, and E because they share a common endpoint.

## 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.

*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: Computing link homophily on a trace graph, $G$.

*Do real trace graphs exhibit link homophily with respect to application layer traffic?* To answer this question, 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 Japan (KEIO). These traces represent a diverse set of network environments, collected at different geographic locations and times. Section IV-A1 describes these traces in detail.

| 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.

*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 endpoint exhibits higher value 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

**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: Neighboring Link Classifier (NLC).

**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: 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 $t$ updates on the graph $G$. The simulated annealing parameters are: $\beta_0 \in [0, 1]$ and $\beta_{t+1} = \beta_t \cdot \alpha$, where $\alpha$ 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.

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 [2] 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.

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: $\forall u \in U : label(u) = argmax_{c \in C}(P(label(u) \equiv c|u))$.

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.

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 [3].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 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 [2]. 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: $\forall u \in U : label(u) = argmax_{c \in C}(P(label(u) \equiv c|u))$.

| | Backbone Traces | | Access Link Traces | |
|---|---|---|---|---|
| | **PAIX** | **WIDE** | **ENTP** | **KEIO** |
| ***Application Traffic Mix*** | | | | |
| **P2P** | 76055 | 893 | 3780 | 79 |
| **Web** | 62860 | 5877 | 88883 | 6868 |
| **DNS** | 39387 | 39532 | 58158 | 16498 |
| **Chat** | 4794 | 734 | 1953 | 140 |
| **Mail** | 6987 | 2973 | 1307 | 2958 |
| **SNMP** | 94 | 9 | 10485 | 52 |
| **FTP** | 152 | 46 | 864 | 2 |
| **SSH** | 18 | 2 | 509 | 7 |
| **Rest** | 9215 | 543 | 1942 | 352 |
| **Unknown** | 47442 | 68946 | 99954 | 8372 |
| ***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.

## IV. Experiments on Traffic Classification

### A. Experimental Design

*1) Data Sets:* We evaluate NLC+RL on four real-world traces from a diverse set of network environments, collected between 2004 and 2007. For internet backbone traces, we have traffic from a commercial US Tier-1 ISP link connecting San Jose to Seattle (PAIX) and another from a transpacific link between US and Japan (WIDE). Our access-link traces are collected at the border router of an enterprise network (ENTP) and from inside Keio University in Japan (KEIO).

Table VI lists the distribution of flow-types for each of the four traces. We define a flow using the well-known 5-tuple $\langle$`srcIP`, `srcPort`, `dstIP`, `dstPort`,`protocol`$\rangle$. 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 [4] and later enhanced in [5]. For each trace, 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.

*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. 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. The size of the LCC depends on the duration of observation. We found NLC+RL's classification performance to be robust to changes in duration of observation.

*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). For our experiments, we *emulate* the existence of a **seed provider**, using a payload-based signature-matching method similar to previous works [6], [4], [7]. 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; and is robust to biases and errors in seed labels. For details, see [1].

*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. To test NLC+RL, 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 Accuracy (ACC) and F1-score (the harmonic mean of Precision and Recall).

### B. Experimental Results

We conducted experiments that answer the following questions. Due to space constraints, we have omitted many details including an in depth discussion; and refer the reader to [1].

**Question** $\#1$**:** 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. See Figure 3.

**Question** $\#2$**:** 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 size over all four traces.

**Question** $\#3$**:** 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.

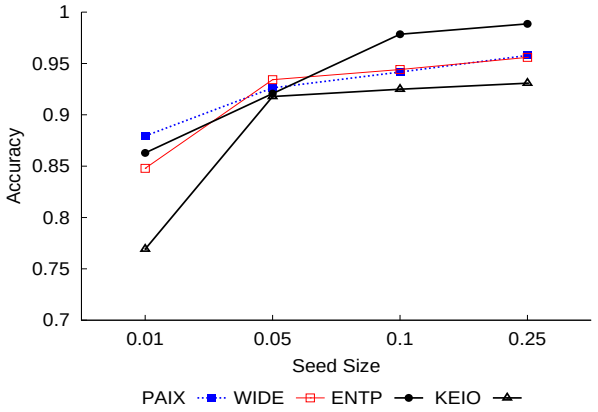


Fig. 3: Accuracy of NLC+RL over all 4 traces.

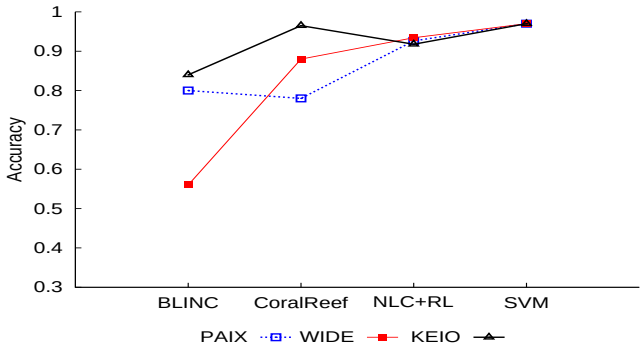


Fig. 4: Comparing NLC+RL with 5% seed size to CoralReef [8], BLINC [4], and SVM [5] on the same traces. CoralReef and SVM rely heavily on port numbers and flow-level data (which can be obfuscated), while NLC+RL does not require such information.

**Question** #4**:** 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.

**Question** #5**:** 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 be associated with different applications.

**Question** #6**:** How does NLC+RL perform compared to other methods? **Answer:** NLC+RL either outperforms or is competitive ($< 4\%$ difference) with existing approaches that rely heavily on port numbers or flow-level data. See Figure 4.

## V. Related Work

Traffic classification is a well-studied problem with significant previous work. Based on the level of observation, traffic classification methods can be divided into four groups: (a) packet-level [8]; (b) flow-level [9], [5]; (c) host-level [4], [10]; and (d) payload-level [6], [11], [12]. To our best knowledge, no one has viewed the traffic classification as a relational learning problem. Graphs have been used to represent network traffic for other tasks besides traffic classification [13], [14], [15], [16]. However, none of the previous works examined link homophily, its presence in real-world trace graphs, and its utilization for traffic classification.

## VI. Conclusions

We observe link 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 and has several attractive features: (a) resistance to signature, padding and timing obfuscation techniques; (b) robustness to errors and biases of the initial seed information; and (c) high accuracy ($> 90\%$) on application classification in real traces.

## Acknowledgements

This work was performed under the auspices of the U.S. Dept. of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 and supported by NSF grants No. CT-0831530, No. NETS-083206, and a CISCO gift.

## References

[1] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos, "Link homophily in the application layer and its usage in traffic classification," Lawrence Livermore National Laboratory, Livermore, CA, Tech. Rep. LLNL-TR-414362, June 2009, http://eliassi.org/papers/bjg-tr09.pdf.

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

[3] 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.

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

[5] 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.

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

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

[8] CAIDA Org., "The CoralReef Project."

[9] 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.

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

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

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

[13] 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.

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

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

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