INTERNATIONAL
COMPUTER SCIENCE
INSTITUTE

A Large-Scale Empirical Analysis of Email Spam Detection Through Transport-Level Characteristics

Tu Ouyang[†], Soumya Ray[†],
Mark Allman and Michael Rabinovich[†]

TR-10-001
January 2010

Abstract

Spam is a never-ending issue that constantly consumes resources to no useful end. In this paper, we evaluate the efficacy of using a machine learning-based model of the transport layer characteristics of email traffic to identify spam. The underlying idea is that the manner in which spam is transmitted has an impact that is statistically observable in the traffic (e.g., in the network round-trip time or jitter between packets). Therefore, by identifying a solid set of traffic features we can construct a model that can identify spam without relying on expensive content filtering. We carry out a large scale empirical analysis of this idea with data collected over the course of one year (roughly 600K messages). With this data, we train classifiers using machine learning methods and test several hypotheses. First, we validate prior results using similar techniques. Second, we determine which transport characteristics contribute most significantly to the detection process. Third, we analyze the behavior of our detectors over weekly and monthly intervals and in the presence of major network events. Finally, we evaluate the behavior of our detectors in a practical setting where they are used in a filtering pipeline along with standard off-the-shelf content filtering methods, and demonstrate that they can lead to computational savings in practice.

1947 Center Street, Suite 600
Berkeley, CA 94704-1198 USA
(510) 666-2900, (510) 666-2956 fax
www.icsi.berkeley.edu

[†] Case Western Reserve University, Cleveland, Ohio

1 Introduction

Spam is clearly an ongoing challenge for both network operators and users. Spam imposes many costs from simple network capacity, computation and storage costs to user productivity issues that arise as people manually filter spam that is not automatically caught or worse, lose legitimate messages in the maze of filters our messages now must traverse. While in general, spam filtering in one form or another works reasonably well in terms of keeping spam away from users without dropping excessive legitimate communication, the costs of this filtering are high *behind the scenes*. For instance, computational complexity issues arise with parsing or tokenizing email, database lookups, etc. Further, depending on the setup the storage requirements of retaining a "junk" folder for users to pick through if they suspect a lost message is also high. For instance, one of the authors' "junk" folders consists of 66 MB of messages over the last 30 days. In addition, spam represents an arms race whereby spammers are constantly developing new ways to circumvent filters (by adjusting their content, where they transmit spam from, etc.) while network administrators are constantly updating their tools and databases to keep spam away from their users. Therefore, while a users' viewpoint might well be that spam appears to be a largely "solved" problem the reality is quite different for administrators and operators.

A number of recent efforts have involved the notion of using network or transport layer features of email traffic to build a model that can disambiguate spam and ham email [20, 6, 14]. For instance, email that comes from bogus IP addresses or does not follow the usual size distribution of email might be more likely to be spam. The overall thrust of these efforts is to try to reduce the cost of spam filtering by either making good decisions without relying on computationally expensive content filters or by making stronger decisions such that (some) spam can be safely thrown away and thus not become a storage burden.

Our work is inspired in particular by [6], which suggested that the way spam is sent causes certain transport-layer artifacts that can be detected, but that are difficult for a spammer to alter in a meaningful way. For example, the act of saturating a link with spam (e.g., from a DSL-connected bot) leads to contention for the scarce uplink capacity and therefore causes metrics such as the number of lost segments and the round-trip time (RTT) to increase. A spammer could send spam at a lower rate to avoid triggering the tell-tale transport features, but this would also be helping to solve the overall spam problem.

While the initial experiment in [6] is promising, our work makes the following key contributions.

- We propose and validate a new methodology for automatic development of ground truth (Section 3). The availability of ground truth is crucual in rigorous evaluation of any classification technique. Yet obtaining ground truth involves the following hard dilemma: large-scale corpora make hand-labeling of objects infeasible while small-scale corpora raise doubts about the representativeness of the results. We believe our methodology for ground truth development will be applicable beyond the scope of our current work and become generally useful in evaluating spam classification techniques.

- We use rigorous experimental methodology in our study (Section 4). We use ten-fold cross validation to evaluate our methods and report true positive and false positive rates, together with Receiver Operating Characteristic (ROC) graphs, which we use to select operating points for practical use of our approach. We perform in-depth feature analysis including an expanded set of transport features. We assess the efficacy of using decision trees for the transport layer models and provide an intuitive interpretation of the resulting classifier.

- We evaluate the utility of transport-level features on a large-scale corpora of email messages collected over an extended period of time. Our data set includes 600+K messages received at ICSI over the course of one year compared to around 18+K messages (with only 200 hams) used in prior work [6]. In particular, our data set allowed us to consider the aging properties of the classification models built

	Sep	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Mar-All**	**Apr**	**May**	**Jun**	**Jul**	**Aug**
Msgs.	377K	467K	279K	316K	291K	241K	245K	1.2M	306K	279K	302K	317K	292K
Outgoing	104K	79K	71K	46K	47K	42K	36K	168K	40K	41K	38K	49K	54K
DNSBL	123K	271K	120K	203K	176K	138K	148K	713K	190K	165K	185K	174K	165K
Unknown	21K	20K	10K	8K	13K	7K	8K	34K	15K	11K	21K	31K	20K
No Msg.	34K	24K	23K	10K	9K	10K	10K	42K	9K	9K	7K	8K	6K
Other Prob.	10K	13K	7K	1K	0K	1K	0K	–	3K	5K	8K	8K	5K
Spam	67K	44K	33K	30K	32K	26K	27K	121K	27K	30K	26K	30K	26K
Ham	18K	16K	16K	18K	17K	16K	18K	80K	22K	18K	18K	18K	15K

Table 1: Dataset characteristics. “Msgs”:Total email traffic, “Outgoing”:Outgoing email, “DNSBL”:DNS blacklist filter, “Unknown”:Invalid user, “No msg.”:No text in email, “Other”:Other problem, “Spam”/”Ham”:Final number of ham and spam we process.

over transport-level features, and their resilience ot major network events (our data spans the McColo shutdown period) (Section 4.2).

- Using a prototype implementation of our approach, we evaluate the deployment of a transport-level classifier proposed in [6] where the transport-level classifier is used as a first-stage filter in front of a content filter (Section 5). By coupling our propotyped classifier with an existing content filter, we are able to concretely quantify savings in processing costs.

Our overall finding in Section 4 is that transport layer features can classify spam with 85–92% accuracy. Further, the transport attributes are fairly stable over time (e.g., the accuracy of an 11 month old model is still roughly 73%). We evaluate individual features to determine their importance, both in isolation and in conjunction with other features, and find that features roughly correlated with the round trip time, sender’s OS and connection bandwidth are relevant to spam detection. Finally, we demonstrate that just using the transport layer classifier to throw away what appears to be clear instances of spam and feeding the rest to traditional content filters significantly decreases the overall resources required to process each message.

2 Data Collection

As part of our general monitoring activities we collect full content packet traces from the border of the International Computer Science Institute (ICSI). Retaining all such traces permanently is prohibitive in a number of ways and therefore the ICSI traces are retained only temporarily—essentially until their operational relevance for forensics and troubleshooting has diminished. For this study we retained packets involving ICSI’s two main SMTP servers.[1] The dataset we use in this paper covers one week each month (the 11^{th} through the 18^{th}) from September 2008 through August 2009. Further, to gain additional granularity we collected data from the entire month of March 2009 as described in the experiments presented below.[2] We only consider mail to ICSI users and not outgoing mail from ICSI users (as this latter would introduce significant bias in the transport characteristics because the two SMTP servers and ICSI’s connectivity would influence them strongly).

ICSI’s standard email setup involves using the Spamhaus SBL, XBL and PBL DNS blacklists [23] to help mitigate the impact of spam. SMTP transactions with backlisted hosts are terminated by the SMTP

[1] We do record email not using one of these two servers, but since in some cases our tools rely on the notion of a well-known mail server to form an understanding of directionality, we exclude this traffic. The excluded traffic is a relatively small fraction of the SMTP traffic recorded. For instance, we exclude 0.3% of the SMTP packets and 0.5% of the SMTP bytes on the first day of our data collection.

[2] Unless explicitly noted data from “March 2009” should be interpreted as the standard one-week of data.

daemon with an error delivered to the remote host. Unfortunately these transactions do not give us (i) any way to understand if the message is actually spam and (ii) a chance to observe a number of our transport level features (described below). So, while ideally we would include these messages in our study it is not operationally feasible to do so within our dataset. Therefore we do not include any connections that are terminated by the DNSBL mechanism in our study. We do note that this setup is an operational reality at many institutions and therefore we do not feel that assuming a DNS blacklist as a first step in the spam filtering process reduces the value of our findings. In fact, filtering off such emails *may* make our job more difficult by removing some of the more egregious sources of spam leaving us to contend with only more subtle spammers.

The characteristics of the data are given in Table 1. The second group of rows denote messages that have been removed from processing in our analysis. Per the above outgoing email and connections terminated based on DNS blacklists both exclude a large number of messages from our corpus. However, a number of additional problematic messages were identified. The "unknown" row indicates messages aimed at an invalid ICSI user—and therefore the message itself is never delivered to ICSI. The cause of these could be a simple typo on a user's part, but it is more likely that most of these come from dictionary-type spamming (i.e., trying to send to bob@icsi, alice@icsi, etc.). Additionally we find many delivery attempts are made without yielding an actual message we can analyze (denoted "No msg." in the table). Finally, there are myriad problems that creep into the process in small numbers and are collected into the "other" category (note, all our data contains such issues, but in some cases they are rare enough that rounding causes the table to list no such errors).

The final two lines in Table 1 show the number of spams and hams in each month to calibrate the reader. The specifics of how we derived these numbers is given in the next section.

From our corpus of packet traces we use *Bro* [18] to re-assemble email messages from each of the incoming connections and *spamflow* [6], *Bro* and *p0f* [25] to generate the transport layer characteristics of each connection. We use the messages themselves to develop ground truth (as described in the next section). Table 2 lists the transport-layer features we distill from the packet traces. Note "local" indicates one of the two ICSI mail servers that are near the packet tracing vantage point, while "remote" indicates the non-ICSI host that is connecting to the ICSI mail server. In other words, the "remote" host is sending mail to the "local" host. Features marked with a "$*$" were derived by *Bro*, those marked with a "¶" were derived by *p0f* and the remaining features were obtained via *spamflow*.[3] Finally, we place a B next to those features that are common between our study and the Beverly [6] study.

The features we use in our analysis were chosen for two basic reasons: (i) because they were used in [6] and had some rough agreement with our intuition and (ii) were aspects of the traffic that our intuition led us to believe might be useful. We tried to be liberal in adding features to our list. Our goal is not to base our entire analysis on our possibly ill-informed mental models, but rather to use our intuition as a starting point. In Section 4.3 we evaluate empirically the efficacy of each feature we have chosen.

We have broken the features into three groups in Table 2. The first category focuses on the nature of the spam itself (although, derived without looking at the content). We include only one such feature: the number of bytes transmitted by the remote host. This feature is included because our general mental model suggests that spam is generally smaller than legitimate email. The second group of features consists of assessments of the remote host itself—e.g., an inference of the operating system. The *ttl* feature is a hybrid of sorts in that it is the residual TTL after traversing the network and so depends on the initial TTL set by the remote host (which varies based on operating system) and the number of hops from the remote to local hosts (which [14] suggests can be used to help predict spam). The final—and largest—group of features listed in the table

[3] Note, we derived these few features outside *spamflow* purely as a convenience and not for any fundamental reason. We believe that all transport layer features could be readily derived in a single place—likely the TCP implementation itself in an operational setting.

Feature	Description
bytecount*	Number of unique bytes transmitted by the remote host.
OS¶	Operating system of remote host.
ttl	IP TTL field from SYN received from remote host.
ws	Advertised window size from SYN received from remote host.
3whs^B	Time between the arrival of the SYN from the remote host and arrival of ACK of the SYN/ACK sent by the local host.
fins_local^B	Number of TCP segments with "FIN" bit set sent by the local mail server.
fins_remote^B	Number of TCP segments with "FIN" bit set received from the remote host.
idle^B	Maximum time between two successive packet arrivals from remote host.
jvar^B	The variance of the inter-packet arrivals from the remote host.
pkts_sent / pkts_recvd	Ratio of the number of packets sent by the local host to the number of packets received from the remote host
rsts_local^B	Number of segments with "RST" bit set sent by the local mail server.
rsts_remote^B	Number of segments with "RST" bit set received from remote host.
rttv	Variance of RTT from local mail server to remote host.
rxmt_local^B	Number of retransmissions sent by the local mail server.
rxmt_remote^B	Approximate number of retransmissions sent by the remote host.
throughput*	The number of unique bytes received from the remote host (*bytecount*) divided by the duration of the connection.

Table 2: Transport features

attempt to capture the "busyness" of the network in a variety of ways. The intuition behind these features is that a bot sending massive amounts of spam may clog the network, increasing loss (retransmissions), RTT, the jitter while decreasing overall throughput. Some previous work [20] has mentioned that individual bots send spam at a low rate. However, their results pertain to a single sink domain and to the overall amount of spam over trace duration rather than the rate over fine-grained time intervals.

Finally, we note that our feature list is not identical to that used in [6]. As indicated in the table, we have added several features not used in the previous study. In addition, we removed three features, as well: *packets* (in each direction), *cwnd0* and *cwndmin*. The number of packets is closely related to the *bytecount* and *pkts_sent/pkts_recvd* features in our list. The two *cwnd* features are actually not about the congestion window, but about the advertised window. The *cwnd0* feature tries to capture the case when the remote host goes into a zero-window probing mode because the local host has exhausted some buffer. Further, the *cwndmin* feature likewise assesses the occupancy of the local TCP buffer. We excluded these because they did not depend on the state of the remote host or the network path. Further, [6] shows these to be not particularly useful in detecting spam. In addition to changing the set of features note that some of the names of the features have been changed to aid clarity.

3 Ground Truth

Assessing any automatic classification technology requires the establishment of ground truth, which is our case means predetermination of which messages are spam and which are ham. Establishing ground truth presents a difficult challenge in large datasets such as ours. Hand-labeling is the best way to establish ground truth as was done for the 18K messages used in [6]. However, there are two problems with using hand labeling to generate ground truth. First, hand labeling does not scale in terms of number of messages (due to the time consuming manual process) or the breadth of messages (due to ethical concerns relating to reviewing email). Second, the definition of "spam" is nebulous and a single person cannot always tell whether a given message is "spam" or "ham". For instance, sometimes a travel advertisement might be unsolicited junk and sometimes it might be the result of the user signing up to receive notifications of particular travel deals. Therefore, hand labeling the 600+K messages from across ICSI is not feasible. Using a single spam filter as in [12] would potentially bias the detection techniques we develop by making our detection mimic a known detection scheme and fall prey to any blind spots present in the given tool. (Note, [12] is not developing new detection methodology, but rather characterizing spam traffic within the context of a working operational setup and therefore the use of a single strategy is less problematic and logistically understandable.)

Consequently, to produce approximate ground truth we developed an automatic labeling procedure that first involves four open source spam filters, each with (largely) independent methodologies for classifying a message. We employed the following tools:

SpamAssassin: We used spamassassin 3.1.7 (current when we started our investigation) [2, 22]. SpamAssassin includes a series of tests and signatures that it uses to score a message—with a threshold then used to determine if the message is spam or ham. We used the default threshold of 5.0. SpamAssassin has a Bayesian learning algorithm and several non-local tests whereby remote databases are queried. We did not use either of these features (in an attempt to stay independent of the other tools).

SpamProbe: We use SpamProbe v1.4b [7] which is a naïve Bayes analysis tool that works on both single and two-word phrases in email. SpamProbe calculates the probability that the given message is spam and classifies anything with a probability of at least 0.6 (the default) as spam.

SpamBayes: We use SpamBayes v0.3 [16] which is another naïve Bayes analysis tool. As with SpamProbe this tool tokenizes the input and calculates the probability a message is spam. While we use two basic Bayesian tools we note that the details of the implementation matter and these tools disagree for 30% of the messages in our January 2009 dataset (as detailed below).

CRM-114: We use the Orthogonal Sparse Bigrams classifier in CRM-114 v20060704a-BlameRobert [1]. CRM-114 is a Bayesian filter that calculates the probability a message is spam using features corresponding to pairs of words and their inter-word distances. We run CRM-114 using the standard defaults except for the "thick_threshold". CRM-114 classifies messages as spam, ham or unsure. While operationally useful, since we are trying to determine ground truth, we force CRM-114 to eliminate the unsure classification and make a decision by setting the threshold to zero.

The latter three tools above need to be trained before they can classify messages. That is, the tools have no built-in notions of ham and spam, but must be given labeled samples of these two classes to learn their characteristics. In our study we used the TREC email corpus from 2007 [9] (the latest available when we started this project). The corpus includes 25K ham messages and 50K spam messages. The messages that comprise the corpus arrived at one particular mail server over the course of three months. A hybrid of automated techniques and hand labeling was used to categorize the messages. This corpus is far from ideal in that the TREC corpus is dated relative to the data we collected for our study. This is problematic because spam is a constant arms race between spammers developing new techniques to evade filters and filters becoming increasingly savvy about spamming techniques. Further, we train each tool one time, whereas in a more realistic setting the tools get regularly trained on new types of emails (either automatically or by being corrected by users, depending on the tool). That said, as illustrated below a manual check of our ground truth indicates that our overall procedure for obtaining ground truth is accurate despite the less than ideal starting point.

An additional note is that the tools chosen do in fact offer differing views on our email corpus. Table 3 shows the pair-wise disagreement between tools used in our study. Even the best aligned tools (SpamBayes and CRM-114) differ in 19% of the cases. This illustrates that we have chosen heterogeneous tools—which we believe is a key to the following procedure for generating ground truth because the tools can effectively check each other.

Tool 1	Tool 2	Disagreement
SpamAssassin	CRM-114	37%
SpamAssassin	SpamBayes	39%
SpamAssassin	SpamProbe	23%
SpamBayes	CRM-114	19%
SpamBayes	SpamProbe	30%
CRM-114	SpamProbe	31%

Table 3: Pair-wise percentage of disagreement between spam detection tools used in developing ground truth.

Our strategy begins with each tool making a judgment on a message and then combine these judgment to derive the overall conclusion. How to use results from multiple classifiers has been the subject of previous work (e.g., [15]), and our initial inclination was to use one of the existing methods, namely, the "majority voting" scheme whereby a message was classified as spam in the ground truth if three or four of the tools judged it as spam. Otherwise, we considered the message ham.

To determine whether our ground truth is sound we manually spot checked a subset of messages from January 2009. Ideally, we would simply have picked a random subset of all messages and verified the classification in our ground truth. However, since we are working with real email the content could be sensitive and therefore we were not able to use an ideal procedure. In particular, we did not manually examine messages that the spam filters universally agreed were ham (with an exception being sketched below) due to concerns over looking through users' email. However, we determined that if at least one tool indicated that a message was spam then the message was likely to not be sensitive and therefore manually looking at a random sample from across the institute was reasonable. Therefore, we chose two kinds of messages for manual checking: (i) those messages that could be—in an automated fashion—clearly determined to involve the third author of this paper (the only author with access to the raw messages) and were marked as ham by all spam filtering tools (this yields 646 of the 35K such clearly ham messages in the corpus) and (ii) a uniformly random sample of 2% of the messages whereby at least one tool classified the messages as spam (this yields 920 of the 47K messages marked as spam by at least one tool in our corpus).[4]

We found that 5 of the 646 "ham" messages to actually be spam. Of the 920 spam-suspected messages selected, we found that 912 were indeed spam. Of these, there were 218 spams that were classified as such by only one tool, 362 spams that were classified as such by only two tools, 250 spams that were classified as spam by three tools and the rest were classified as spam by all four tools. This showed us that using a majority vote criterion would result in unacceptably high false negative rates (about $218/912 = 23.9\%$, even after breaking ties in favor of spam). On the other hand, consider the following strategy: classify an email as spam if at least one tool says it is spam, and classify it as ham otherwise (i.e. if all tools agree it is ham). This strategy has an $5/(912 + 5) = 0.55 \pm 0.07\%$ estimated false negative rate (FNR, i.e. spam classified as ham), and a $8/(646 - 5 + 8) = 1.23 \pm 0.11\%$ estimated false positive rate (FPR, i.e. ham classified as spam). The overall estimated error rate is then $13/(917 + 649) = 0.83 \pm 0.09\%$. To verify that this is not an artifact of the samples being chosen from January 2009, we then carried out similar tests using messages from September and May and obtained comparable results—(0.3% FNR, 2.39% FPR) for September and (0.28% FNR, 1.56% FPR) for May. Therefore, based on our samples, this strategy appears very accurate.

In retrospect the any-one approach is intuitive in that off-the-shelf spam filters are conservative in classifying a message as spam because the consequence of a false positive (blocking a legitimate mail) is more severe than that of false negative (allowing a spam to reach the user). This causes the errors to go in the direction of identifying less spam. The manual checking shows that being more aggressive in combining

[4] Note, we used both incoming and outgoing messages as the basis of our collection and therefore the numbers do not directly align with those in Table 1.

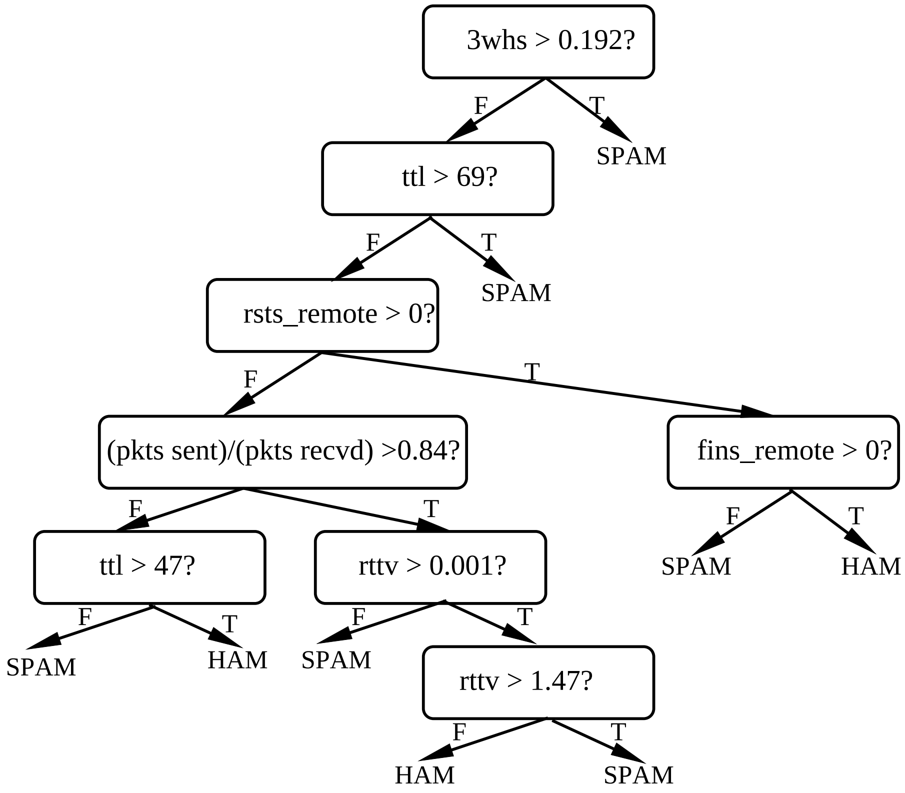

Figure 1: Fragment of a decision tree learned from the January 2009 data.

the filters' judgments can mitigate this effect and reduce the error significantly, which is important since the cost of inaccuracy is symmetrical when determining ground truth (as opposed to an operational setting).

We note that we do not use the results of the manual inspection to train our classifier. In fact, the messages chosen for manual inspection were removed from the remainder of the analysis in our experiments. Further, note that we are developing ground truth by observing *content features*. Our experiments in the following sections involve classifying messages based on *transport features*. We believe that even if there is a small error in our ground truth that error does not bias our results because the content and transport features are (significantly) independent.

4 Empirical Evaluation

We evaluate several hypotheses in our work. First, we hypothesize that transport characteristics of a message provide a good indication of whether a message is ham or spam. Further, we also hypothesize that these characteristics stay stable over periods of time. Finally, we hypothesize that using transport characteristics as a filter before using a content-based approach will lead to a savings in processing time as compared to using a content-based filter by itself. Along with these hypotheses, we also design experiments to evaluate the effectiveness of individual transport characteristics when used in isolation or together with other characteristics.

To test our hypotheses, we construct predictive models from our data and evaluate them according to several performance criteria. The models we chose for our work are *decision trees* [17], which are commonly used in machine learning. Decision trees work using the idea of *recursive partitioning*: at each step, they choose a feature (for us, this is a transport characteristic) and use it to partition the data. This is repeated until a partition only has examples from one category (for us, until we get a partition consisting only of ham or spam), which becomes a leaf node annotated with that category. To classify an example, it is "threaded" down the tree by checking each test until it reaches a leaf. At this point it is assigned the category associated with this leaf. Decision trees have several advantages for our purposes. First, they are computationally very efficient: learning a tree takes on average $O(mn \log m)$, where m is the number of examples and n is the number of features, and classification is just $O(d)$, where d is the depth of the tree. This is important for us because we are working with large data sets. Second, decision trees are easy to understand for people. Once constructed, we were able to inspect the trees and derive insights applicable to our problem. Third, it is easy to modify the tree construction procedure to incorporate *differential misclassification costs*, that is, to produce trees that take into account that mistaking ham for spam is in general a worse error than mistaking spam for ham. Finally, preliminary inspection of our data revealed that some of our features have "threshold effects." Suppose for a feature f, if $f \leq x$, a message is more likely to be spam; if $x \leq f \leq y$ a message is more likely to be ham, if $f \geq y$, a message is more likely to be spam. In such a case, a decision tree with successive tests for $f \leq x$ and $f \geq y$ will easily lead to the desired classification.

To learn decision trees from our data, we used the implementation of decision trees in the machine learning toolkit Weka [24]. This implementation is called "J48." We ran this with the default parameter settings of "-C 0.25 -M 2" in all cases except where noted. The parameter C denotes a confidence threshold used for pruning the decision tree after construction and the parameter M denotes the minimum number of examples in any leaf of the tree (i.e., once the number of examples reaches this value, the partitioning process stops). These parameters have not been tuned to our application. It is possible that improved results might be obtained if these parameters were tuned further. A part of a decision tree learned on one month's data is shown in figure 1 (we discuss this tree in more detail in Section 4.1).

To evaluate the performance of our tree models, we measure several metrics using stratified 10-fold cross validation (implemented in Weka). This cross validation procedure is standard machine learning methodology and produces 10 independent test sets that are *stratified* based on the class label, i.e. the proportions of ham to spam in each fold is the same as the overall proportion. In each iteration, we learn a tree using J48 on nine folds, then evaluate it on the held out fold (the "test set"). We report the average value of several metrics on the 10 folds. First, we report predictive accuracy on the test set. This is a general measure of agreement between the learned trees and the ground truth (as established in Section 2). Further, we report the true positive rate (TPR), which is the fraction of spam that was correctly identified as such by our models. We also report the false positive rate (FPR), which is the fraction of ham that was erroneously classified as spam. Note that we treat spam as the positive class. Finally, we construct Receiver Operating Characteristic (ROC) graphs by thresholding a probability measure reported by J48. This probability indicates, for each prediction, the "confidence" of the classifier in the prediction. To plot an ROC graph, we threshold this confidence measure and plot the TPR against the FPR at each point. This graph gives us a detailed view of the behavior of the classifier and allows us to choose a suitable operating point that trades off an acceptable false positive rate against the true positive rate. As a summary statistic for the ROC graph, we report the "area under ROC" (AUC), which has been shown to correct several weaknesses of accuracy as a metric [19].

4.1 Performance of Transport Layer Characteristics

Our first experiments test the hypothesis that transport characteristics of a message provide a good indication of whether a message is spam. To do this, we perform 10-fold stratified cross validation using data from each month separately and report the averaged metrics outlined above. These results are reported in Table 4. ROC graphs associated with September 2008 and January 2009 are shown in Figure 2.

From the table, we first observe that it is indeed the case that transport characteristics provide a reasonable discrimination between spam and ham. With our classifiers, for each month, we obtain accuracies between 85% and 92%. Note that the majority class predictor, that always predicts "spam," has an average

Month	Accuracy	TPR	FPR	AROC
September 2008	0.931	0.964	0.194	0.938
October 2008	0.898	0.942	0.222	0.917
November 2008	0.885	0.919	0.185	0.919
December 2008	0.892	0.920	0.155	0.927
January 2009	0.876	0.908	0.182	0.914
February 2009	0.869	0.897	0.176	0.907
March 2009	0.877	0.898	0.154	0.917
April 2009	0.890	0.899	0.120	0.931
May 2009	0.880	0.902	0.157	0.917
June 2009	0.868	0.891	0.165	0.906
July 2009	0.883	0.910	0.164	0.917
August 2009	0.886	0.913	0.160	0.917

Table 4: Cross validated accuracy, true positive rate (TPR), false positive rate (FPR) and area under ROC (AROC) of decision tree models for each month's data.

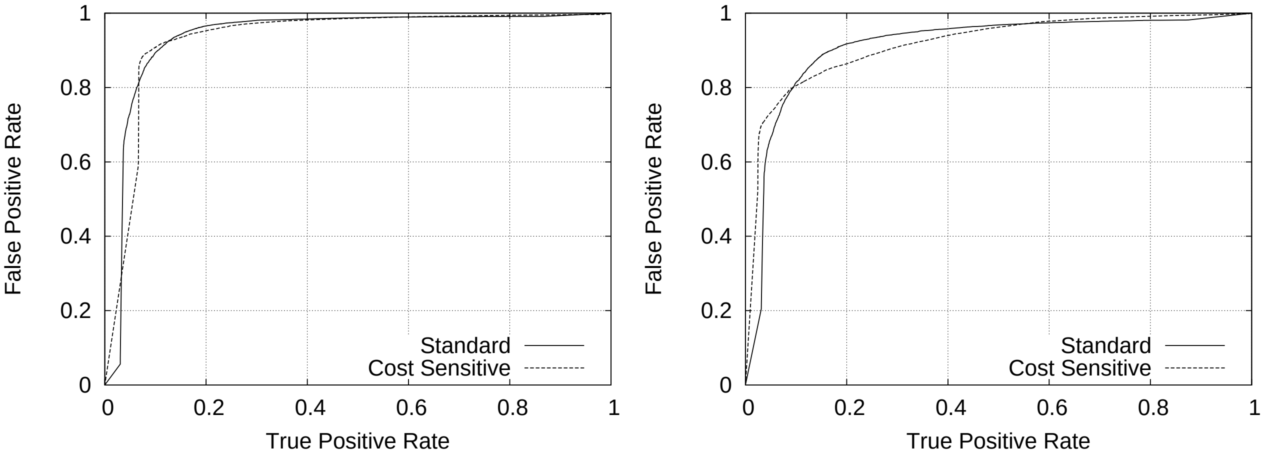

Figure 2: ROC graphs for standard and cost sensitive trees for September (left) and January (right). Note that while the cost sensitive trees sometimes have lower performance overall (lower AROC), they have higher TPR at low FPR.

accuracy of about 50% to 75%. The upper bound is an artifact of September 2008, as McColo was shut down soon after and the fraction of spam drops significantly. This result is in agreement with initial findings [6]. We note that the true positive rate is very high in each case, indicating that most spam can be detected via the transport characteristics. However, the false positive rate is also quite high for these classifiers, indicating that ham emails also occasionally share similar transport characteristics. Because of this, it would not be practical to use this classifier as the only basis of spam detection in a real system. We address this issue below using a *cost-sensitive* classifier. In Section 5 we present a realistic evaluation of this system as a pre-filter for a content classifier.

In figure 1, we show a fragment of a decision tree obtained when using the data from January 2009. In this tree, each leaf is labeled with the "majority class", i.e., the category associated with the majority of messages in that leaf. The most important discriminant in this tree is the length of the initial 3-way handshake. The importance of this feature has also been noted previously [6]. We have also observed that the constant 0.192 seconds associated with this test seems quite stable across different trees. A possible hypothesis to explain this might be a correlation with the geographical origin of spam; this is something we are currently investigating. The next feature in the tree in a *ttl* feature, which is an indication of how many router hops a message has gone through. This may seem like a counter-intuitive feature to use near the root of the tree. However, we found that the default *maxTTL* varies across operating system: Windows XP and related operating systems use an initial TTL of 128, while Linux uses 64. Accordingly, in our data, the *ttl* feature is strongly correlated with the operating system, and in fact provides a finer grained discrimination because of its continuous-valued nature. So we believe that a test for $ttl > 69$ is effectively a determination between Windows or Unix-based remote hosts. If this test is true, then the originating machine was likely a Windows-based machine and the message is likely a spam [5]. The next test checks for a reset in the connection. If there was a reset from the remote host to terminate the connection we find it likely this message is spam; if not, it is more likely to be ham. If there was no reset, the tree looks at ratio of outgoing to incoming packets. The expected case is for this to be around 0.5 due to delayed acknowledgments [5] which calls for one ACK to be sent for every second data segment received (unless a second data segment does not arrive within a small amount of time). However, additional factors can move this ratio towards 1, including (i) some operating systems acknowledgment strategies that transmit more ACKs early in the connection to help open the congestion window, (ii) reliance on the delayed ACK timer to strobe ACKs into the network because of small windows and long RTTs and (iii) because smaller transfers don't have many opportunities to aggregate ACKs. We find that with SMTP traffic and small email messages the ratio tends towards 1 for this last reason. The SMTP control traffic is symmetric (with requests and responses) and therefore has a ratio of 1. When the message is small there are few data packets and therefore coalescing ACKs is either non-existent or done in small enough numbers that the overall ratio is still closer to 1 than to 0.5. In our January data we find that when the ratio is closer to 1, 85% of the messages are below 6K in size and all but 1 are less than 11K in size. On the other hand, if the ratio is closer to 0.5, there is a wide distribution of message sizes, with several messages over 100KB. Therefore, the ratio turns out to be largely a proxy for message size. When the ratio tend more towards 0.5, the tree checks the *maxTTL* feature again. If this indicates that the message has gone through multiple router hops (low TTL), it is more likely to be spam, else if is more likely to be ham. For the "small" messages, the tree then checks the round trip time variance. This feature behaves in an interesting way. If it is too small, this suggests the message had very few packets (since they will then arrive in quick succession and be more likely to experience similar network conditions) and so is likely to be spam. For example, for January 2009 data, of the examples that reach the decision tree node with the $rttv < 0.001$ check and have this small variance, 85% are 6K or less, all but one are less than 9K and 75% are spam. If this feature is too large (there is some delay between packets and also a lot of variation in the delay), this may indicate the connection experienced congestion and is likely to be spam as well. In the "normal" range—there is some delay, but not too much variation in the delay—the message is likely to be ham.

As mentioned above, though the accuracy is reasonable, the FPR for our initial classifiers is too high for practical use. This can be remedied in two ways. First, for a given classifier, we can use its ROC graph to select an operating point with an acceptable FPR. At this point, the TPR will be lower than the maximum possible; however, this might be acceptable if for example we plan to use this to pre-filter messages before deeper analysis methods are used. This method does not alter the learned classifier. We can also improve the FPR if we incorporate into the learning process the fact that false positives (i.e., ham classified as spam) are less desirable than false negatives (i.e., spam classified as ham). This can be done through *differential misclassification costs*. Here, we specify a cost matrix that tells the learning algorithm how much each type of error will be penalized. In Weka, we run the CostSensitiveClassifier implementation using a cost matrix that specifies a cost of 10 for a false positive error against a cost of 1 for a false negative. This cost matrix is then used by J48 when learning the tree (as before, these costs have not been tuned for this application). The cross validated performance of the resulting trees are shown in Table 5. The ROC graphs for these trees for

[5] It is important to remember that the presented tree is just a part of the full tree, which contains over 1000 nodes. The full tree does not judge *all* email messages with $ttl > 69$ to be spam, although the *majority* of such messages are spam.

Month	Accuracy	TPR	FPR	AROC
September 2008	0.888	0.877	0.072	0.932
October 2008	0.826	0.786	0.063	0.922
November 2008	0.796	0.712	0.033	0.905
December 2008	0.806	0.707	0.029	0.929
January 2009	0.790	0.693	0.030	0.917
February 2009	0.783	0.663	0.027	0.907
March 2009	0.801	0.684	0.023	0.915
April 2009	0.809	0.662	0.017	0.927
May 2009	0.810	0.712	0.028	0.920
June 2009	0.772	0.634	0.030	0.902
July 2009	0.795	0.694	0.034	0.920
August 2009	0.796	0.698	0.037	0.919

Table 5: Cross validated accuracy, true positive rate (TPR), false positive rate (FPR) and area under ROC (AROC) of *cost-sensitive* decision tree models for each month's data, where the ratio of false positive cost to false negative cost is 10:1.

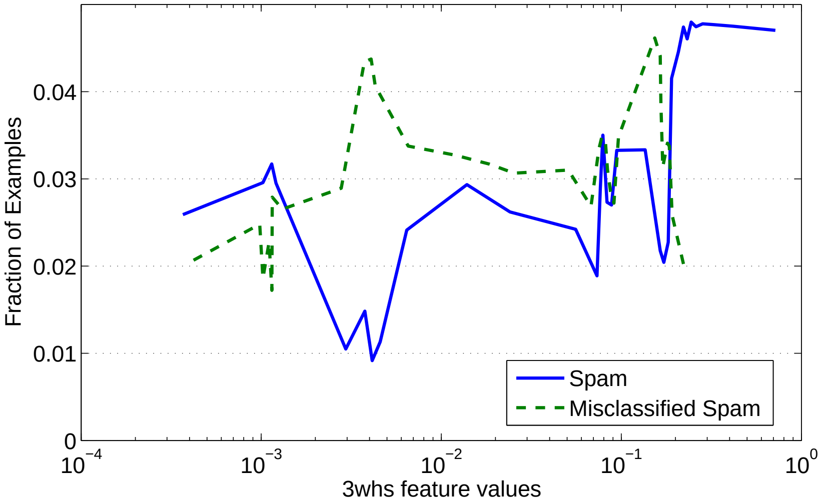

Figure 3: Frequency histograms of the *3whs* feature for spam messages in January 2009 and misclassified spam in January 2009. One outlier with *3whs* value exceeding 100 was deleted for ease of presentation.

September 2008 and January 2009 are plotted in figure 2. From these results, we observe that, as expected, the false positive rate has dropped significantly. Of course, the false negative rate has risen and consequently, the accuracy has dropped. However, from the ROC graphs we note that these trees have higher true positive rates at lower false positive rates. As mentioned before, we use these results to choose an operating point with an acceptable tradeoff in a filter pipeline. This is discussed in more detail below in Section 5.

It is useful to ask if there is a way to characterize the set of examples that are misclassified by the trees learned by this process. To do this, we constructed histograms that represent the distribution of values of each transport feature in (i) all ham and spam messages and (ii) those ham and spam messages that were misclassified by the learned trees in January 2009. We then ran a correlation measure between the histograms corresponding to each feature, to discover how similar they were between the two groups. We show an example of the two histograms for the feature "3whs" in figure 3 and a table of correlations in table 6. Using this analysis, we found that for the spam that is misclassified, six features have low or negative correlation with the other spam messages. In other words, these six features look very unlike

Feature	**Ham**	**Spam**
idle	0.077	−0.376
3whs	−0.042	−0.543
jvar	0.139	−0.309
OS	0.775	−0.174
bytecount	0.159	−0.501
throughput	0.283	−0.498

Table 6: Feature correlation between ham misclassified as spam by cost-sensitive trees and all ham (**Ham** column), and for spam misclassified as ham and all spam (**Spam** column), for January 2009.

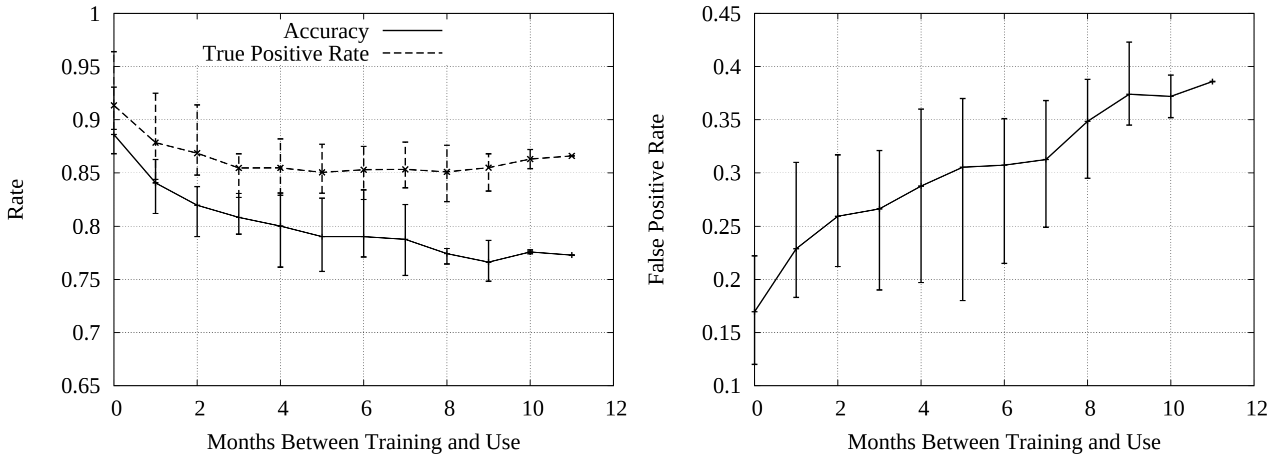

Figure 4: Change in accuracy and TPR (left) and FPR (right) over a year for our models.

the features for spam in general. These six features are *3whs, idle, jvar, OS, bytecount* and *throughput*, several of which are among the most discriminating features in our study (this is described in more detail in Section 4.3 below). We found five of the six features to also have low or negative correlation between the misclassified ham messages and the rest. What is the significance of these features? Roughly speaking, these features identify three properties: the bandwidth of the connection, the OS of the sender and the size of the message. Our intuition is that while we expect most ham (or spam) to have similar properties in terms of these characteristics, there will always be a fraction that will not. Thus there will always be, for example, some ham that originates from Windows machines on low bandwidth connections and some spam that originates from Linux machines on high bandwidth connections. So the feature distributions in our misclassified messages are an indication of the limitations of using just network features to discriminate between ham and spam. To get these classifications correct, we will need to perform a deeper analysis of these messages, perhaps using content-level features.

4.2 Stability of Transport Layer Characteristics

In this section, we describe experiments that test the hypothesis that transport characteristics stay stable over periods of time. To do this, we produce a decision tree using all of each month's data. We then evaluate these trees on each successive month, for example, the tree produced for January 2009 is evaluated on data from February through August 2009. For the month of March 2009, we have data for each week. For this month, we also construct a tree for each week and test it on the following weeks. In each case, we plot a graph

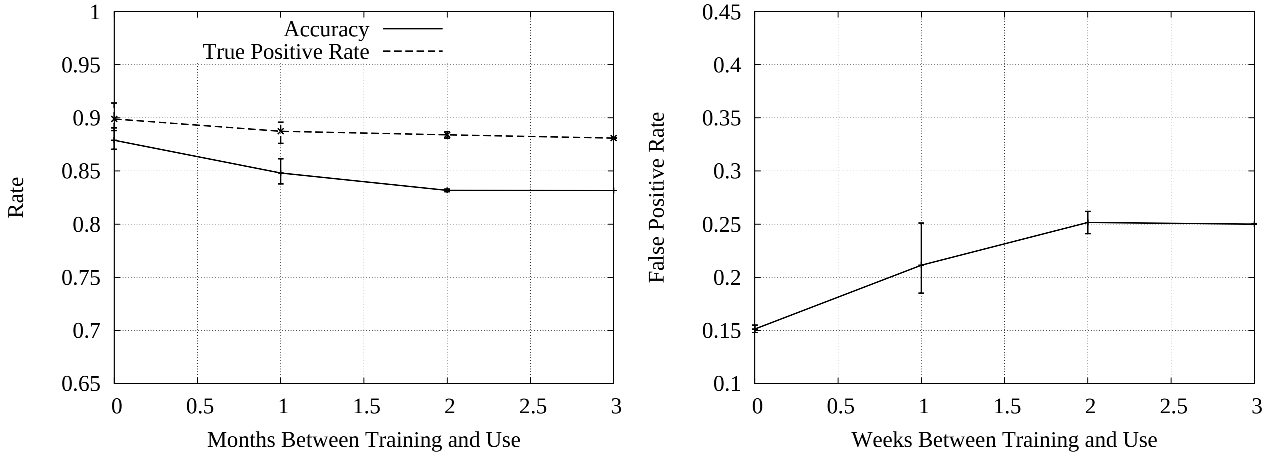

Figure 5: Change in accuracy and TPR (left) and FPR (right) over four weeks in the month of March 2009 for our models.

where the x-axis represents the difference in months (or weeks) between when the tree was constructed and when it was used. For each x, the y-axis then averages the performance over all month pairs that differ by x. Thus for an x value of 2, we average the performance for the trees built in September and tested in November, built in October and tested in December, and so forth. $x = 0$ represents the average cross-validated performance over the year (or month for the weekly experiment). The results are summarized in figures 4 and 5.

From these results, we observe that, as may be expected, the accuracy degrades slowly over time. However, the TPR is very stable across long periods of time and the FPR is increasing, which partly explains the degradation in accuracy. The same pattern is seen over the four-week period in March, although, as might be expected, to a lesser degree. There are two possible explanations for the increasing FPR. First, it might be that the flow characteristics of ham drift over long time periods, so that it becomes increasingly difficult to classify ham correctly as time goes by. Alternatively, it is possible that the FPR is simply tracking the overall ham-to-spam ratio: in months where there is more ham, the FP rate is higher. To check this, in figure 6 (left) we plot the average correlation across features between the ham and spam of each month to the global distribution (i.e., the distribution of each feature when all messages across the year are collected together). From this figure, if we ignore the first three months (this was the "McColo" period, discussed below), we observe that the features of ham are very stable across time. Thus it is not likely that the changes in FPR reflect drift in the network characteristics of ham. We then plot the change in the fraction of ham between pairs of months and compare that to the FPR in figure 6 (right). We observe that the FPR tracks the change in the fraction of ham reasonably well; this seems to suggest that this is the explanation for the variable FPR we observe. From figure 6 (left), we further observe that while spam in any month exhibits lower correlation to the global distribution than ham, the variability of the flow characteristics of spam is still limited: this may be why the TPR of these models stays stable over long periods of time. However, the variability that exists appears to have an interesting cyclical pattern. We do not know yet if this pattern has a systematic explanation.

A key network event during the past year was the shutdown of McColo. McColo was a a hosting service provider that hosted master controllers for several large botnets. On November 11, 2008, McColo's two upstream ISPs cut them off and so the command and control of the botnets was lost. This resulted in a sharp drop in the overall volume of spam on the Internet. To study how significantly email flow characteristics might have been changed by this event, we plot the results for our models constructed with data from

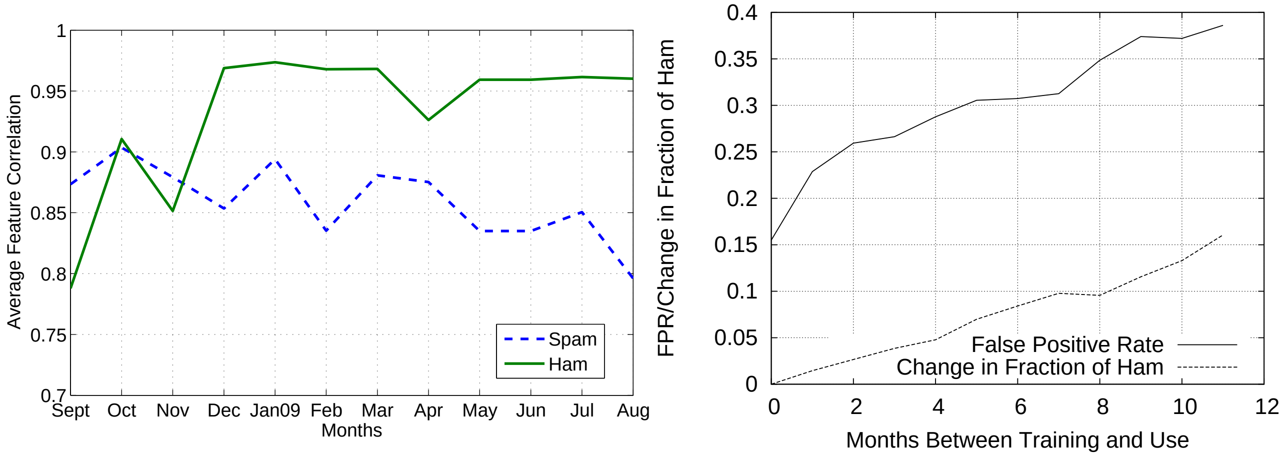

Figure 6: **Left:** Feature correlation between ham for each month against ham overall and spam for each month and spam overall. **Right:** False positive rate and change in the fraction of ham messages over time.

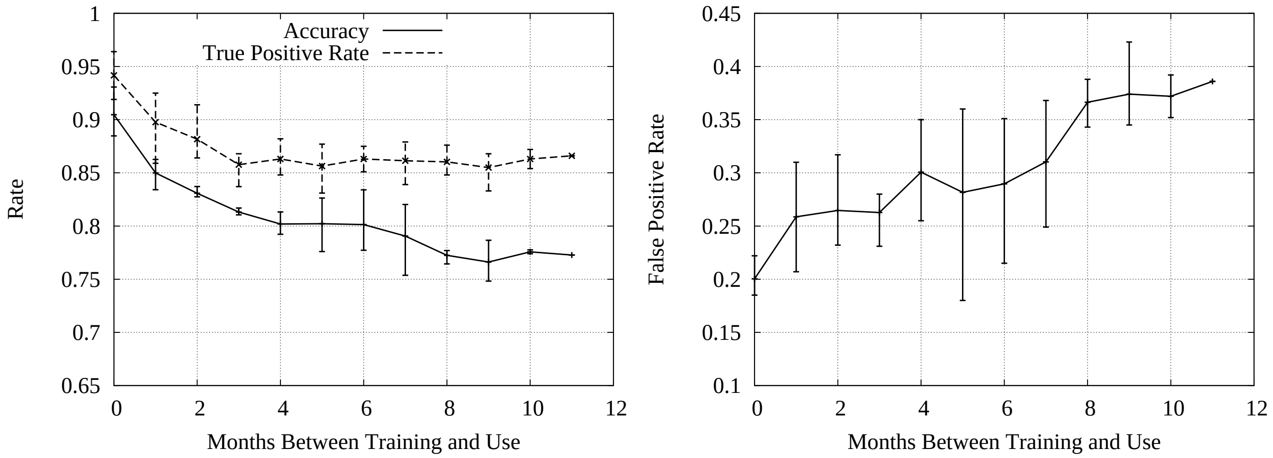

Figure 7: Change in accuracy and TPR (left) and FPR (right) over a year for models constructed with data from September through November 2008.

September to November 2008 separately in figure 7. From these results, we observe that our models are in fact very robust, even over such events. While there is an increased degradation of accuracy and TPR, over the long term the performance is similar to that in Figure 4. The FPR of these classifiers shows a more discontinuous increase than in the overall case: not only is it affected by the changing fraction of ham, as can be see from figure 6 (right), but the flow characteristics of ham messages change somewhat after November 2008, which likely contributes to the rapid rise in FPR after three months.

4.3 Utility of Transport Layer Characteristics

In this section, we report the results of experiments that test which transport layer characteristics are useful in discriminating ham from spam when used in conjunction with other features. To evaluate features in isolation, we produce decision trees using just that feature, and record their cross-validated performance on each month. To evaluate features in conjunction with other features, we perform "lesion studies." Here, for

Feature	**Accuracy**	**TPR**	**FPR**
Full Tree	0.886	0.914	0.170
Empty Tree	0.638	1.000	1.000
3whs	0.777	0.881	0.426
ttl	0.765	0.858	0.422
idle	0.753	0.843	0.440
jvar	0.710	0.831	0.542
pkts sent/recvd	0.695	0.883	0.604
throughput	0.690	0.852	0.633
bytecount	0.687	0.809	0.575
ws	0.682	0.853	0.667
rsts_remote	0.682	0.846	0.932
rttv	0.677	0.931	0.807
OS	0.675	0.737	0.508
fins_local	0.659	0.951	0.867
rxmt_local	0.649	0.969	1.000
rsts_local	0.642	0.994	0.644
fins_remote	0.641	0.957	0.948
rxmt_remote	0.641	1.000	0.672

Table 7: Average accuracy, TPR and FPR when only a single feature is used to construct a tree. The first row shows the averages for the full tree and the second shows the values for the empty tree.

each month, we leave one feature out and record the cross-validated performance of the tree constructed with the remaining features. Thus, these experiments measure if a feature "adds value" to the discriminating ability of the trees, *in the presence of* the other features. Notice that a feature that is valuable in isolation may not be as valuable in conjunction with other features, because the quantity it measures might be revealed equally well by a combination of other features.

The results of using each feature in isolation is shown in Table 7. The first row shows the average cross validated accuracy, TPR and FPR for the full trees across all months, and the second row shows these quantities for an empty tree, that always simply predicts the majority class. Notice that for this tree, TPR and FPR are both 1. The following rows show these quantities when only a single feature is used, sorted by accuracy. From this table, we observe that several of our features, including *3whs*, *ttl*, *idle* and *jvar*, are good discriminants between spam and ham. Other features, such as *rsts_remote* and *rttv*, are mediocre discriminants, while yet others, such as *rsts_local* or *rxmt_remote* are very poor discriminants, basically yielding the empty tree if used in isolation. Even though some features appear mediocre in isolation, however, they may still be useful in a tree, where they are *conditioned* on the values of other attributes. Thus, as we showed in figure 1, *rsts_remote* is a useful feature in the tree, when it is conditioned on certain values of *3whs* and *ttl*, though it is not very useful by itself.

The results of the "leave-one-feature-out" lesion studies are shown in figures 8 and 9. In these figures, we plot, for each feature, the average change in accuracy, TPR and FPR if that feature is deleted, along with 95% confidence intervals. From these figures, we first observe that no single feature is critical to the success of models: even leaving out features such as *3whs* do not affect the accuracy of these models by a very large margin in absolute terms. This indicates that our choice of features is such that they capture overlapping aspects about the data: even if one feature is deleted (or somehow not measured for a message), the remaining features are able to pick up most of the slack. Though this is the case, however, we also observe that for certain features, deletion results in a statistically significant decrease in accuracy. The features *3whs* and *ttl* belong to this category. Further, we observe that though for every feature, deletion results in a small negative change to the accuracy, though these changes are not statistically significant. In other words, none of our features are really "noise" features that are irrelevant to the problem; so on average, we do not expect to improve accuracy by removing any of the attributes we currently have.

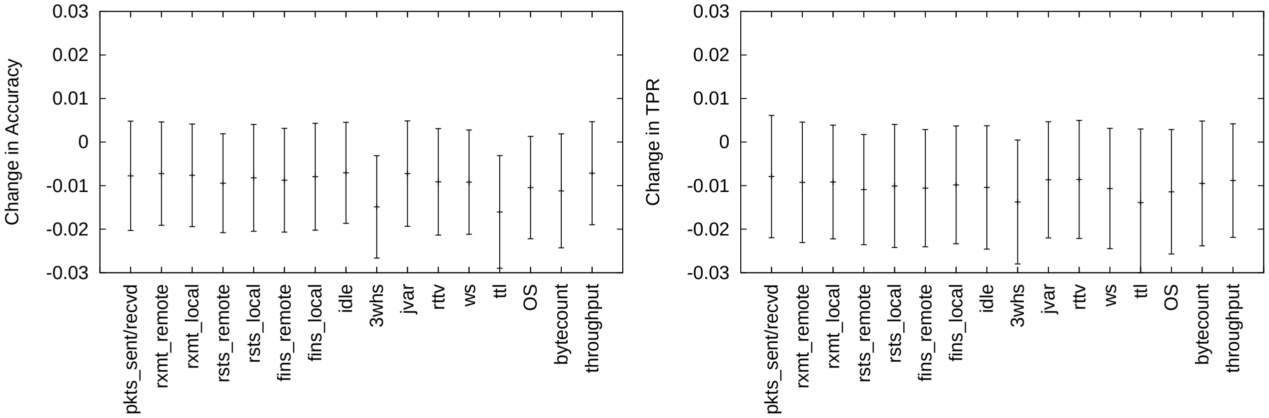

Figure 8: Change in accuracy (left) and true positive rate (right) as each feature is left out of the set of features and the tree is constructed.

5 Operational Use

As a final experiment we turn to assessing the use the methodology developed in this paper in an operational setting. We have not actually implemented and deployed the following strategy, but roughly assess the components to illustrate that the approach is promising in real-world terms.

Given the results in the previous sections, a filter based on transport-layer characteristics is not accurate enough to use as the sole arbiter of email messages. However, we can use a filter based on our developed model as a first pass at filtering. Messages that are determined to be spam in this first-pass filter are not further processed. However, messages determined to be ham are further analyzed with a content filter. Our goal is to reduce the number of messages that must be analyzed by heavyweight content filters. For this experiment, we used the data from December 2008 to develop a model learned with false positive cost of 100. We then used its ROC graph to choose an operating point with a false positive rate of roughly 0.6%. At this point, its true positive rate was 42%. We then use this model to classify messages from January 2009. This process identifies 14K of the 48K messages as spam. We found that all but 90 of these predictions were correct. Since in this month there were 31K spam and 17K ham, the TPR of this pre-filter for January is 45% and FPR is 0.5%, in line with our expectations. The TPR is lower than the results presented in previous sections due to our tight constraint on the false positive rate.

Transport Classification: The process of classifying messages based on transport characteristics has two components that we consider in turn: (i) deriving the given features and (ii) evaluating those features within a particular model.

To determine the effort required to calculate transport features we consider the length of time *spamflow* requires to process our packet traces.[6] This process requires 114 seconds for the January 2009 dataset.

Next we need to understand the complexity of executing the ML-based decision tree model across each of the messages in our corpus. We used Weka to directly analyze all 49K messages from January 2009 in bulk and the process takes 2 seconds, which illustrates the low-cost of the procedure (even if bulk processing is infeasible in an operational system). In addition, we translated the decision tree produced by Weka into a Python script, which is then executed for each of the 49K messages in our corpus. This process takes 783 seconds. This is clearly a fairly high upper bound on the amount of time required for this classifi-

[6] All the times given in this section were computed on the same machine. The absolute magnitude of the numbers is not important and would be different on another host. Our goal is to explore the relationships between different processing techniques.

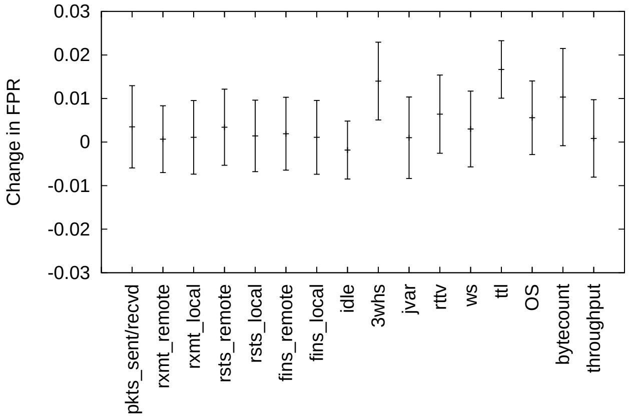

Figure 9: Change in false positive rate as each feature is left out of the set of features and the tree is constructed.

Spam Filter	**Content-Only** (sec)	**Transport + Content I** (sec)	**Transport + Content II** (sec)
SpamAssassin	3,074	3,143 (-2%)	2,362 (23%)
SpamProbe	1,390	2,001 (-44%)	1,220 (12%)
SpamBayes	9,609	7,778 (19%)	6,997 (27%)
CRM	4,895	4,389 (10%)	3,608 (26%)

Table 8: Real-world simulation results.

cation. In particular, we executed a Python script that did nothing 49K times and that process consumed 746 seconds—illustrating that most of the time (95%) is consumed by our classification script is in the overhead of starting and stopping the process and not in the complexity of the classification task.

In our analysis below we use two different measures of the cost of the ML-based analysis to illustrate the bounds of the implementation space. Our expectation is that a working system with kernel-level tracking of transport characteristics in a TCP implementation coupled with queries for this information and execution of the model with the MTA will leave the real cost closer to two seconds than to 783 seconds. However, for the purposes of this paper we wish to show the entire spectrum. Therefore in option I we sum the 114 seconds required to derive the transport layer characteristics and the 783 seconds required to execute the Python script 49K times, or 897 seconds as the cost of the ML-based classification In addition, we calculate the cost in option II as the sum of the 114 seconds required to derive transport layer characteristics and the two seconds required by Weka to bulk classify the 49K messages.

Content Classification: Next, since our goal is to save effort over running all messages through heavyweight content filters, we assess the time required by each of the four content filters we used in Section 3 to develop ground truth. We assess both the time required for the entire 49K message corpus as an estimate of the current spam filtering cost. In addition, we asses the time required to analyze only the 35K messages the transport-layer classification cannot make.

Table 8 shows the results of our analysis. The second column gives the total amount of time required to use each of the content filters to process all 49K messages in our corpus. The third and fourth columns show the pessimistic and optimistic amount of time required by considering spam filtering to be a two-stage process with the first stage winnowing 14K messages from the set that requires content-based analysis. In all cases except two we see improvements of at least 10% and more than 20% in many cases. The results show that SpamAssassin[7] and SpamProbe actually fare worse with a pre-processor if we take the upper bound on the length of time required by the transport classifier (i.e., 897 seconds). However, using a lower bound assumption on this task (i.e., 116 seconds) the pre-filter can improve the performance of even efficient tools.

We note that assuming the transport model is used to discard messages and not further process them is not the only way to effectively use transport-layer classification. The classification is likely cheap enough that it could be used to prioritize messages. I.e., messages the transport characterization identifies as spam could be run through content filters during slow times when the server has plenty of spare capacity, whereas non-spam could be further checked as soon as possible. Another possibility is that transport layer judgments could be used to "backup" a content-based judgment and if both identify a message as spam then it could be discarded completely, whereas if only one classified a message as spam the message could be retained in the users "junk" folder such that users can catch mistakes. Additional possibilities no doubt exist, as well.

In addition to time savings, it is possible that using a transport filter improves the system's overall accuracy, in particular because the transport filter may be able to detect spam that a content filter misses. This observation was also made in prior work [6]. We have done preliminary experiments that suggest this is true; however, we do not have conclusive results yet. This is an immediate direction for future work.

6 Related Work

Spam detection and mitigation has attracted significant attention within the industrial, operational, open source, and research communities. Spam filtering plays an important role in these efforts. There have been a vast range of filtering techniques proposed and implements including (with examples cited rather than an exhaustive list): DNS blacklists [23]; email address whitelists based on a user's address book in many popular email clients including GMail, Thunderbird and Outlook [3]; domain signed messages [4], analyzing transport [6] and network layer [20, 14] characteristics and signature [2, 22] and statistical [13] analyses of a message's content.

Content filters are quite popular—both within an institution's mail processing apparatus and in users' client software—and classify a message using the information in the message itself. Content filters have been shown to be very effective in combating spam, but are computationally expensive and require constant re-training as spam content adapts in an attempt to evade detection [8, 10]. Content filters are also less effective with non-textual spam, although efforts have recently emerged to address this limitation [11].

Given the high cost of content-based filtering (as illustrated in Section 5), several efforts have attempted to use network-layer information to detect spam flows. In particular, Ramachandran and Feamster analyze network-level characteristics of spamming hosts [20], and Hau et al. exploit network-level features, such as the sender's geodesic distance, autonomous system (AS), etc. for spam detection [14]. Schatzmann et al. propose a collaborative content-blind approach that an ISP can deploy to facilitate spam blocking, based on size distribution of the incoming SMTP flows [21]. The idea is that SMTP servers that perform prefiltering terminate spam flows quickly after processing the envelope; hence flow size from these servers can be used to rate client MTAs for the benefit other SMTP servers that might not use prefiltering. The flow-based approach is complimentary to these techniques, in that it focuses on non-content features in the traffic.

The idea of using transport features was originated by Beverly and Sollins [6]. Besides validating their general approach using a larger dataset, we also develop a method for automatic generation of ground truth, examine additional transport features, investigate different machine learning techniques (decision trees—which are far less computationally expensive and therefore more attractive to real-world use), consider data that spans one year and evaluate a strawman usage scenario.

[7] Note, we used the SpamAssassin daemon in these experiments. The running time without the daemon is approximately an order of magnitude longer. In addition, as when using SpamAssassin to produce ground truth we did not use non-local nor statistical tests. Therefore, SpamAssassin's running time should be taken as a lower bound.

7 Conclusions and Future Work

In this work, we have conducted a large scale empirical study into the effectiveness of using transport level features to detect email spam. Though the idea has been presented in prior work, to the best of our knowledge, our work is the first thorough exploration of the various issues involved. We conduct our study using data collected over the course of a year. Because hand labeling such a large corpus is impractical, we develop a methodology for constructing sound ground truth within the context of a large email corpus that is both too big and sensitive to hand label. Our method depends on four spam tools that use different assessment strategies. Using manual validation of a sample of email we find the error in the ground truth to be roughly 2%. Using this data, we explore several hypotheses. First, we validate prior results on our corpus. We discuss the significance of the feature tests in our tree models, address the issue of false positives and establish the limits of flow-based email classification. We then consider the stability of flow features and find that they remain stable over long periods of time and even to some extent across major network events. Next, we consider the individual features in our data and establish that several features capture important aspects of the data; however, they overlap so that no single feature is critical to our model, which contributes to its robustness if a feature should be unmeasured or noisy. Further, there does not appear to be any truly irrelevant features in our model. Finally, we develop a possible use-case that involves using transport layer-based filtering as a first step in the overall spam detection process. By using this first step to eliminate a fraction of the messages before passing the remainder on to more expensive content filtering schemes we show an overall savings in the amount of work required to process incoming messages. While there are certainly additional ways to leverage transport layer classification, our initial use-case shows the efficacy of the additional information.

There are a number of directions we intend to investigate in future. First, as mentioned in the previous section, we intend to assess the overall performance of the two-stage filter we have sketched in terms of classification ability. In addition, we intend to investigate the degree to which we can more tightly integrate transport layer and content layer filtering. That is, rather than considering these as two distinct steps in a process we hope to leverage key features in both dimensions, possibly adding network-level features, to produce stronger and more efficient email classifiers. Finally, we plan to implement the developed approach and deploy it in a real setting.

References

[1] Crm114 - the controllable regex mutilator. `http://crm114.sourceforge.net/`.

[2] SpamAssassin. `http://spamassassin.apache.org/`.

[3] Whitelist Instructions. `http://sci.scientific-direct.net/wl.html`.

[4] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas. Domain-Based Email Authentication Using Public Keys Advertised in the DNS (DomainKeys), May 2007. RFC 4871.

[5] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control, 2009. RFC 5681.

[6] R. Beverly and K. Sollins. Exploiting Transport-Level Characteristics of Spam. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008)*, Aug. 2008.

[7] B. Burton. SpamProbe. `http://spamprobe.sourceforge.net/`.

[8] G. Cormack and A. Bratko. Batch and online spam filter comparison. In *The Third Conference on Email and Anti-Spam*, 2006.

[9] G. Cormack and T. Lynam. 2007 TREC Public Spam Corpus. `http://plg.uwaterloo.ca/~gvcormac/treccorpus07/`.

[10] G. V. Cormack and J. M. da Cruz. Using old spam and ham samples to train email filters. Available at http://www.j-chkmail.org/ceas-09/gvcjm-ceas09-full.pdf; a short version appeared at CEAS'2009, 2009.

[11] G. Fumera, I. Pillai, and F. Roli. Spam filtering based on the analysis of text information embedded into images. *Journal of Machine Learning Research*, 7, 12 2006.

[12] L. H. Gomes, C. Cazita, J. M. Almeida, V. A. F. Almeida, and W. M. Jr. Characterizing a spam traffic. In *Internet Measurement Conference*, pages 356–369, 2004.

[13] P. Graham. A Plan for Spam, 2002. `http://www.paulgraham.com/spam.html`.

[14] S. Hao, N. A. Syed, N. Feamster, A. G. Gray, and S. Krasser. Detecting Spammers with SNARE: Spatio-temporal Network-level Automatic Reputation Engine. In *Proc. USENIX Security Symposium*, 2009.

[15] T. R. Lynam, G. V. Cormack, and D. R. Cheriton. On-line spam filter fusion. In *SIGIR*, pages 123–130, 2006.

[16] T. Meyer and B. Whateley. SpamBayes: Effective Open-Source, Bayesian Based, Email classification System. In *Proc. First Conference on Email and Anti-Spam (CEAS)*, June 2004.

[17] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[18] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the 7th USENIX Security Symposium*, Jan. 1998.

[19] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 445–453, Madison, WI, 1998. Morgan Kaufmann.

[20] A. Ramachandran and N. Feamster. Understanding the Network-Level Behavior of Spammers. In *ACM SIGCOMM*, 2006.

[21] D. Schatzmann, M. Burkhart, and T. Spyropoulos. Inferring spammers in the network core. In *PAM*, pages 229–238, 2009.

[22] A. Schwartz. *SpamAssassin: The Open Source Solution to SPAM*. O'Reilly, 2004.

[23] Spamhaus DNS Blacklists. `http://www.spamhaus.org/`.

[24] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

[25] M. Zalewski. p0f: Passive OS Fingerprinting tool. http://lcamtuf.coredump.cx/p0f.shtml.