NEVERMIND, the Problem Is Already Fixed: Proactively Detecting and Troubleshooting Customer DSL Problems

Yu Jin*, Nick Duffield†, Alexandre Gerber†, Patrick Haffner†, Subhabrata Sen†, Zhi-Li Zhang*
*Computer Science Dept., University of Minnesota †AT&T Labs–Research
*{yjin,zhzhang}@cs.umn.edu †{duffield,gerber,haffner,sen}@research.att.com

ABSTRACT

Traditional DSL troubleshooting solutions are reactive, relying mainly on customers to report problems, and tend to be labor-intensive, time consuming, prone to incorrect resolutions and overall can contribute to increased customer dissatisfaction. In this paper, we propose a *proactive* approach to facilitate troubleshooting customer edge problems and reducing customer tickets. Our system consists of: i) a *ticket predictor* which predicts future customer tickets; and ii) a *trouble locator* which helps technicians accelerate the troubleshooting process during field dispatches. Both components infer future tickets and trouble locations based on existing sparse line measurements, and the inference models are constructed automatically using supervised machine learning techniques. We propose several novel techniques to address the operational constraints in DSL networks and to enhance the accuracy of NEVERMIND. Extensive evaluations using an entire year worth of customer tickets and measurement data from a large network show that our method can predict thousands of future customer tickets per week with high accuracy and significantly reduce the time and effort for diagnosing these tickets. This is beneficial as it has the effect of both reducing the number of customer care calls and improving customer satisfaction.

1. INTRODUCTION

Digital subscriber line (DSL) is a key technology that brings the Internet to our homes through dedicated copper lines via the local telephone network. Given the importance of the Internet to our daily life, we want and expect high-reliability and high-quality DSL network service. Unfortunately, DSL service problems can and do occur. Most such problems occur at the network edge beyond the DSL Access Multiplexer (DSLAM, see Section 2 for the architecture of DSL networks) down to and including customer premises wiring and home equipment issues, which we refer to as *customer edge problems* in this paper. When encountering such a problem, getting it diagnosed and resolved can be a frustrating experience for both service providers and customers.

The state-of-the-art operational approaches for detecting and troubleshooting customer edge problems utilized by DSL service providers are mainly *reactive* in nature. Customer agents wait for a subscriber to report any problem encountered and use domain knowledge to match the symptoms described by the customer to determine if a technical problem occurs at the customer edge. In order to locate the network element causing the problem, a dispatch is often scheduled and a field technician runs a truck roll, going physically to a customer's home and/or other locations, and checking numerous devices one by one using a plethora of meters and tools until the problem is finally solved.

Not only is such a reactive process time-consuming and labor-intensive from the ISP side, it also diminishes the experience for the customer. When a customer calls the technical help line, she is usually asked to go through a series of routine diagnosis steps such as removing all filters, rebooting the DSL modem, desktop, etc. Such an interview serves as a preliminary to identifying the root of the problem. However, this interview at times may not be successful and the customer will then be asked to schedule with field technicians a physical dispatch to the customer premises for the problem to be detected and resolved. It may take one or more days (due to physical scheduling of people and equipment) until the network is back to normal. A lengthy resolution can lead to customer dissatisfaction and ultimately lead to churn, i.e., customers terminating their contracts.

Instead of relying on a reactive solution, *can we troubleshoot customer edge problems more proactively and with substantially greater efficiency?* This is the central challenge which this paper attempts to address. We would need to identify customer edge problems and fix them quickly before the customers complain about them or even realize the existence of these problems. If successful, we can thereby reduce the number of trouble tickets, achieve faster resolutions and potentially reduce churn. Our goal is to develop

ACM CoNEXT 2010, November 30 – December 3 2010, Philadelphia, USA.

techniques that work within the confines of the existing operational environment, utilize only existing information infrastructure and are not predicated on expensive new infrastructure investments. Although these conditions impose constraints, they are vital for ensuring the successful deployment and adoption of resulting techniques by the operators.

In this paper, we propose a novel *pro-active* approach – referred to as *NEVERMIND* – for troubleshooting customer edge problems. NEVERMIND is designed with two interrelated goals: i) to detect potential customer edge problems in advance, or, equivalently, to predict future customer tickets, and the operators can then resolve them before customers call to complain or even before they realize the problems; and ii) to assist technicians in quickly locating and troubleshooting problems, thereby speeding up the diagnosis process. Both goals target improving the customer experience on the DSL network by greatly reducing the current lengthy troubleshooting process of customer edge problems. Correspondingly, NEVERMIND consists of two components: a *ticket predictor* which detects potential customer edge problems which may lead to future customer tickets, and a *trouble locator* which prioritizes potential problem locations to assist technicians in diagnosing problems.

The key challenges in designing NEVERMIND lie in the practical operational constraints. Given the complexity of DSL networks, especially at the customer edges, there exist many *passive* simple physical components at different locations (inside a customer's home, the switch box outside the home, along the phone line to switching office, etc.), which may cause the problem. They are dedicated devices specific to a customer and cannot be remotely "pinged" or queried for their status. Therefore, it is very difficult to obtain fine grained status measurements for these devices, unless a technician physically goes to test them on site. The lack of such fine grained data renders many existing network troubleshooting methods proposed in the literature, e.g., [3–5, 9, 10, 12], no longer applicable, as most of them are developed for active network elements, or rely on correlations of various statistics collected on these active elements. In comparison, NEVERMIND does not require fine-grained and continuous measurements. Instead, it utilizes existing *sparse and noisy* line measurements collected weekly at a fixed location in the DSL network (via a few remotely accessible devices, e.g., DSL modems and DSLAMs), which reflect the *aggregate status* of all (passive) devices along each individual dedicated copper line. Such line measurements contain metrics regarding physical layer line conditions, such as the bit rate, the estimated loop length and so on. Mining such sparse and noisy data manually is a daunting task. Employing advanced machine learning algorithms, NEVERMIND automatically mines inference rules by correlating periodic line measurements with existing customer trouble tickets which contain diagnosis provided by field technicians. This enables NEVERMIND to identify accurately potential customer edge problems and accelerate the troubleshooting process even with such imperfect measurement data.

Limited operational resources are another key factor that affects the design of NEVERMIND. In order to seamlessly integrate NEVERMIND into the existing DSL troubleshooting architecture, a high priority would be assigned to customer reported problems, with the remaining operational capacity used by NEVERMIND. To make best use of the remaining capacity, NEVERMIND ranks predicted problems by their likelihood of occurrence so that the operators would focus on the most likely cases. For this study, we consider the top ranked 20K problems predicted by NEVERMIND.

In order to make accurate predictions, i) we devise a novel *top-N average precision based feature selection method* to select a compact line feature set, which enables the ticket predictor to yield high accuracy within the capacity of the operation resources without sacrificing the scalability of the system; and ii) we put forth a *combined model* which incorporate the hierarchical structure of typical DSL networks. This combined model can help technicians efficiently locate problems even when a problem only occurs rarely in the past.

We evaluate NEVERMIND using line measurement data, customer tickets and other data sources collected from a major DSL network in the US in 2009. This network contains millions of subscribers across many different geo-locations. Evaluations show that, using the ticket predictor, we can accurately predict more than 8,000 future tickets per week with high accuracy. Investigation of the predictions that did not lead to tickets shows that they are also likely related to customer edge problems but somehow not reported by DSL customers, e.g., a customer is not aware of the problem because she was not using the DSL service when the problem happened (see Section 5.2). In addition, during the dispatches for resolving these predicted tickets, the trouble locator can significantly improve the speed for locating problems.

The remainder of the paper is organized as follows. Section 2 overviews DSL networks and points out several challenges for troubleshooting customer edge problems. In Section 3, we discuss the problems in the current reactive DSL customer care solution and propose our proactive solution – NEVERMIND. The details of ticket predictor in NEVERMIND are explained in Section 4 and the evaluation results are presented in Section 5. We briefly discuss the trouble locator in Section 6. The related work is introduced in Section 7 and Section 8 concludes the paper.

2. BACKGROUND AND CHALLENGES

In this section, we introduce the architecture of typical DSL networks and study the characteristics of customer edge problems in DSL networks. Finally, we point out several challenges in troubleshooting such problems.

2.1 Introduction to DSL Networks

DSL is a widely-deployed technology for supporting digital signal transmission over traditional telephone lines. Fig. 1 shows the architecture of a typical DSL network. At the

customer's premise, customer computers and other network devices are connected to a DSL modem which in turn is connected to a phone line. This dedicated connection terminates at a DSL Access Multiplexer (DSLAM). Each DSLAM typically terminates the per-subscriber dedicated DSL connections for several tens of customers. It separates voice from data signals, sending the former to the Plain Old Telephone Service (POTS) infrastructure. The DSLAM aggregates data traffic from the customers and feeds it to an upstream ATM switch. The latter extracts IP traffic (the traffic from the DSL modem to the DSLAM is typically ATM) and sends it to a Broadband Remote Access Server (BRAS) server which is the gateway from the DSL access network to the IP core network.The above describes commonly deployed DSL infrastructure and some variations and evolutions exist. For example, newer IP-DSLAMs perform the ATM to-IP conversion internally obviating the need for an external ATM switch.

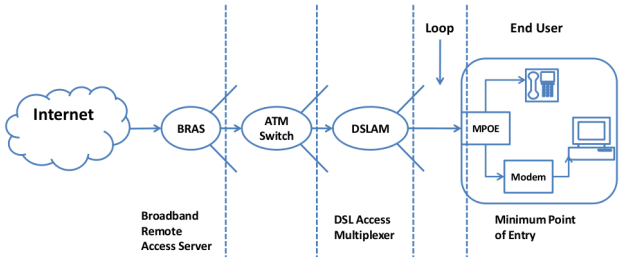


Figure 1: DSL network architecture

2.2 Problems in DSL Networks

A customer edge problem only affects a particular customer, since all the elements on a dedicated DSL line are specific to each customer. In contrast, due to the hierarchical structure of a DSL network, a problem occurring between a BRAS server and a DSLAM pair usually affects the connectivity of multiple customers served by this pair. Operators refer to such problems as *network outage problems*.

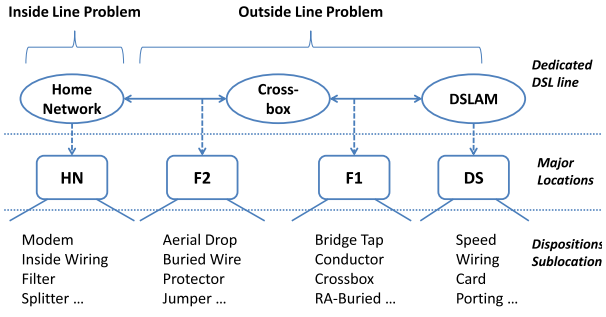


Figure 2: Root causes of customer edge problems.

Earlier research has mainly focused on troubleshooting network outage problems. However, customer edge problems form the overwhelming majority of all problems occurring in DSL networks. In comparison to existing works, in this paper, we focus primarily on predicting and troubleshooting customer edge problems. In addition, as we shall see in Section 5, by correlating predicted customer edge problems based on geolocations, we can also provide useful information to the operators about potential outage problems at DSLAMs. In the following, we briefly discuss the plausible causes of different customer edge problems.

A dedicated DSL line connects a DSL modem at the user end to the DSLAM serving the nearby area. There are a large number of components on each DSL line or inside customer's home network (e.g., inside wiring or equipment) that may contribute to a reported customer edge problem. To understand the different causes of customer edge problems, we study the customer tickets received in August, 2009 (see Section 3.3 for details of all the datasets). Each ticket is associated with a disposition note from the field technician indicating the problem resolution action - either a device or component (e.g., a defective DSL modem or a worn cord) had to be fixed/replaced, or a configuration change effected (e.g., stabilizing the line by downgrading speed). We summarize the representative dispositions and their corresponding locations in Fig. 2 and provide examples of common problems and dispositions in Table 1.

These dispositions can be partitioned into four major categories (or major locations) based on the places where the problems are resolved: the home network (`HN`), the DSLAM (`DS`), the path between the crossbox and `DS` (`F1`), and the path between `HN` and the crossbox (`F2`). Field technicians today break up the end-end path into these four locations, and troubleshoot by individual location. For example, if the technician has enough evidence to believe a problem happens at `DS`, she can save time by skipping testing other three locations. `DS`, `F1` and `F2` can be also clustered into a single larger class - *outside (home network)* problems. However, due to the fact that there is no dominant disposition in these major locations (Table 1), it is difficult to make such a decision purely based on expert knowledge.

2.3 Challenges

The large number of possible dispositions coupled with very limited instrumentation on the real-time health of the network makes troubleshooting customer edge problems a difficult and expensive task. Automated troubleshooting proposals, e.g., [4,10], require continuously tracking the health of different network components and then inferring the trouble locations through the correlations across these components. Such approaches are not well-suited to our problem, where there exists substantial heterogeneity in terms of elements and their capabilities, and to what extent they can be monitored. Most elements on a DSL line (e.g., cable and splitter) are basic physical devices. To measure the state of such devices, a technician needs to be on site at the location of the device and manually run diagnostic checks using specialized tools to determine the current state of the device. Due to the huge number of DSL subscribers, continuously

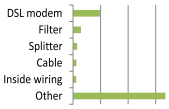
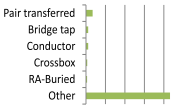
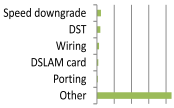
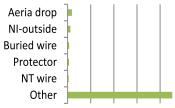
Location	Disposition (pct.)	Description
HN		Defective DSL modem issues Filter issues Splitter issues Network cable issues Inside wire (wet, corroded, cut, etc.) Jack, software, NIC, etc.
F1		Transfer service to another cable pair bridge tap of the customer's facilities Wet or corroded wire conductor The defect is found in a crossbox Defective buried ready access terminal Pair cut, defect cable, stub, etc.
DSLAM		Reduce speed to stabilize the line Digital stream transport Wiring at DSLAM DSLAM pronto card ABCU/ADLU Porting Digital stream, ATM switch, etc.
F2		Aerial drop was replaced Access point (DEMARC) - Outside Repaired existing buried service wire Defect in protector unit Wire from protector to DEMARC Jumper, defective MTU, etc.

Table 1: Dispositions at different major locations.

monitoring the states of millions of devices via such manual testing is infeasible. Even where it is possible to probe the state remotely (e.g., for DSL modems), operational constraints and resource limitations restrict the fidelity and frequency of these reports. We note that many of these constraints are a function of the limitations of existing deployed technologies and current best practices in operation. Making changes to deployed infrastructure, even if possible, can be prohibitively expensive and hence we need solutions that work within the constraints of today's operational environments.

3. CURRENT SOLUTION AND NEVERMIND

In this section, we introduce today's widely used reactive DSL troubleshooting approach and point out the problems with such an approach. Motivated by these problems, we propose the system NEVERMIND, which seeks to aid proactive resolution of customer edge problems.

3.1 Current DSL Troubleshooting Solution

The classical approach to troubleshooting customer edge problems, widely used in most DSL networks, is a reactive process. It relies on customers to report any problem encountered, and then dispatches technicians to resolve each problem using their domain expertise. Fig. 3 (the top box) shows the architecture of such a solution. The customer agents wait for customers to report any customer edge problem by phone and interact with the customer to perform some basic diagnosis (e.g., rebooting the DSL modem or bypassing the filter) to filter out as many non-technical problems as possible, such as billing issues, etc. If the problem cannot be solved by the customer agents, the ticket will be escalated to ATDS (Automatic Testing and Dispatching System). ATDS combines various sources of historical data (e.g., any construction/rewiring recorded in the nearby area) and online testing (e.g., results of metallic check and other online testing tools from the telephone company) to determine whether the problem can be resolved by remote configuration. If not, ATDS will contact the customer to make an appointment and a field technician is sent to the end host to locate the problem manually. After the dispatch, ATDS either closes the ticket or arranges a second-round dispatch depending on the feedback from the customer.

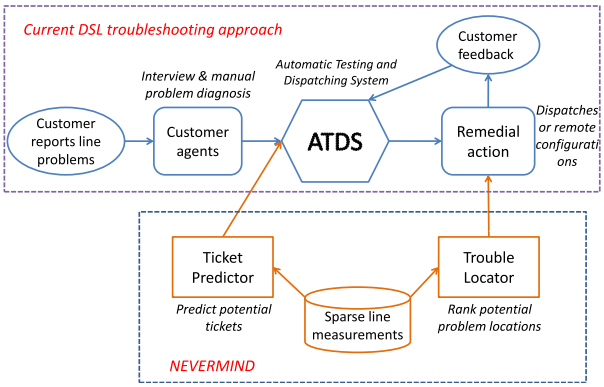


Figure 3: The current DSL troubleshooting approach and the proposed NEVERMIND approach.

The current approach has various drawbacks. Operators rely on customer tickets and interaction with customers to guide them towards discovering the underlying problem and solution. However this is very challenging. Most customers quite naturally lack the technological expertise to pinpoint the actual problem. They typically are able to describe the observable symptoms that caused them to call. Unless the underlying problem and its symptoms are well-known, the operator does not have enough quality information to lead the customer through questions to do effective root-cause-analysis. These factors contribute to limiting the effectiveness of the interview, and most tickets end in a determination of a field dispatch, often with an inaccurate prognosis about the likely cause. A large number of such dispatches would be unnecessary or at least more effective if better quality information was available. For example, if a problem is caused by a defective DSL modem, this can be solved by placing an order for a new modem device online without an expensive field dispatch. Further, waiting for the customers to report customer edge problems will implicitly lead to increasing users' dissatisfaction with the service. In order to have a problem fixed, a customer needs to wait in the queue for an available agent to report the problem and may talk to several agents until a dispatch is scheduled. It may take one or more days (due to physical scheduling of people and equipment) for the problem to be finally solved by field technician. When a problem is not correctly identified and resolved (e.g., intermittent connection problems), the customer needs to call multiple times to have it fixed. All these prolong the

troubleshooting process and lead to large number of unnecessary customer tickets, which is a noticeable contributor to the increase in churn.

Another problem with the current solution is that it depends too much on the experience of the field technicians. A more experienced technician will be able to select an appropriate ordering of locations to test, which enables her to more quickly find the actual problem earlier on; while an inexperienced technician may waste time in testing far more possible locations before identifying the site of the problem.

3.2 Overview of NEVERMIND

In this paper, we propose NEVERMIND, a proactive solution for troubleshooting customer edge problems in DSL networks, which can be seamlessly integrated into the existing DSL troubleshooting framework. The architecture of NEVERMIND is illustrated in Fig. 3 (the bottom box). It consists of two main components, a *ticket predictor* and a *trouble locator*. The ticket predictor utilizes existing sparse line measurements of individual DSL lines to proactively identify lines for which customers are likely to register problem tickets in the near future. These predicted tickets are then submitted to ATDS to be diagnosed in a similar way as normal customer tickets. However, we shall point out that due to the high complexity of the diagnosing process employed by ATDS, we need to set high priority to customer reported problems, and only utilize the remaining operational resource (after resolving customer reported problems) for NEVERMIND. The number of predicted tickets that can be handled daily by ATDS is usually upper bounded by a few thousand in our testing DSL network. For both types of tickets, dispatches are arranged if necessary. Before a dispatch, the trouble locator creates a list of all possible trouble locations ordered by the likelihood of being the cause of the problem. The list is then sent to the field technician to accelerate the troubleshooting process.

NEVERMIND depends on extracting useful information by joining various existing data sources. In the following, we first introduce the various information sources employed by NEVERMIND for making accurate inferences.

3.3 Information Sources

1. DSL Line Measurements: These contain the results of periodic remote line tests initiated from DSLAM servers to each of the connected DSL lines. The line test is conducted as follows. Every Saturday, each DSLAM server initiates connections with the DSL modem on each DSL line and exchanges a few packets with the modem. Based on this conversation, several *metrics* or *line features* are computed to reflect the current condition of that DSL line. We summarize these 25 line features in Table 2.

These line features were defined by the operators based on their collective domain expertise and experience gained from multiple years of operating the service, and represent the dimensions they have observed to be most useful for determining the health of the network and for troubleshooting problems. These features were originally used to assist the customer agents identify customer edge problem tickets from other types of tickets. For example, based on predefined manual rules, an agent will escalate the customer ticket to ATDS if either the current bit rate is lower than the minimum bit rate indicated by the profile, or the relative capacity is greater than 92%. Expert knowledge indicates that such line status is more likely caused by a defective device, and hence it may require further diagnosis by field technicians. Though limited, there are also a few manual rules used by the technicians today to determine the disposition of a particular problem. For example, an estimated loop length greater than 15,000 ft often indicates that the current customer profile is not supported by the DSL line, and a speed downgrade is necessary to reduce the noise on the line. However, due to the high dimensionality of the feature space and unknown/latent relationships between customer edge problems and these features, manually deriving accurate inference rules is very difficult. As we shall show later, our proposed solution involves an effective method for identifying such relationships and using that knowledge effectively for ticket prediction and trouble location.

Feature	Description
state	if the modem is on
dnbr, upbr	bit rate (kbps)
dnpwr, uppwr	signal power
dnnmr, upnmr	noise margin
dnaten, upaten	signal attenuation
dnrelcap, uprelcap	relative capacity
dncvcnt1, dncvcnt2, dncvcnt3	code violation interval counts with different thresholds
dnescnt1, dnescnt2	the number of seconds in which code violations occurred
dnfeccnt1	downstream forward error correction counts with value not less than 50
hicar	the biggest carrier number
bt	the existence of a bridge tap
crosstalk	the existence of cross talk
looplength	estimated loop length
dnmaxattainfbr, upmaxattainfbr	maximum attainable fast bit rate
dncells, upcells	rolling count of cells

Table 2: Basic line features. Prefixes "dn" and "up" means downloading and uploading, respectively.

The once-a-week occurrence of these measurements limits the fidelity of this information source, and arises from operational considerations of minimizing any impact on normal customer activities (hence run on weekends) as well as historical perceptions of how often they needed to be run (was originally considered part of maintenance procedures). From the arrival time of customer tickets received in August, 2009, we observe a clear weekly trend, where the number of tickets peaks on Monday and hits the bottom over the weekend. Since the line tests are conducted on a weekend, we can afford to reserve more resources to resolve the predicted tickets proactively during these relatively "quiet" periods,

and thereby also reduce the new ticket load during the week.

2. Customer Trouble Tickets: These contain problems reported by customers (e.g., slow speed or no connection) and corresponding solutions provided by customer agents (e.g., rebooting the modem or replacing the cable).

Customer agents assign a coarse label to each ticket, indicating the category that the ticket belongs to. We differentiate customer edge problem tickets from other types of tickets such as billing related issues based on this label.

3. Ticket Disposition Notes: These are created by field technicians and serve as a summary for each individual dispatch. Each note contains the ID of the corresponding customer ticket and information regarding the dispatch, such as the starting time, duration, etc. Though the field technicians do not record the details of the diagnosis process such as results from intermediate testing steps, they do provide a *disposition code* to indicate which device was finally identified to be cause of the problem, e.g., a defective modem or a worn cable, or how the problem was ultimately resolved, e.g., stabilizing the line performance through downgrading the line speed. Such disposition codes are used as our ground truth for studying each problem. We note that the disposition codes are determined based on the expert knowledge of the technicians and hence can be very noisy. If a problem is caused by multiple devices, the code is always associated with the device closest to the end host.

4. Subscriber Profiles: These specifies the parameters (or expected values of the line features) for individual DSL lines, which depend on the type and level of service that a customer has subscribed for. For example, DSL customers with the basic profile are expected to have a downloading rate of 768kbps and an uploading rate of 384kbps. In comparison, customers with the advanced profile will have a downloading rate of 2.5Mbps and an uploading rate of 768kbps.

All these datasets are from one of the major DSL providers in the US, with a population of millions of DSL subscribers across multiple geolocations. The datasets are collected by multiple DSLAMs during the whole year 2009. Typically these datasets are stored at different locations and are brought together for our work[1]. We note that even though the specifics of data may vary for different networks and service providers, our system can be applied readily for proactively troubleshooting problems in other operational networks.

4. PREDICTING TROUBLE TICKETS

In this section, we introduce the ticket predictor, whose function is to select a number of DSL lines that are likely to have future customer reported tickets and submit them to ATDS for further diagnosis. This problem is equivalent to a ranking problem, where the ticket predictor selects down millions of DSL lines to a smaller set where problems may happen. The decision of the ticket predictor is based on existing sparse line measurements. Therefore, the key in designing the ticket predictor is to identify the correlation of line measurements and customer tickets.

There are two factors affecting the design of the ticket predictor. First of all, the weekly DSL line measurements are quite sparse, i.e., only a maximum of 52 records are available for each DSL line over a whole year period, hence we can only construct very coarse grained time-series for individual DSL lines, which do not provide us sufficient details for extracting meaningful patterns. Therefore, the challenge is how to convert these time-series into meaningful features. After that, we employ advanced machine learning algorithms to automatically learn inference rules to predict future tickets using these features. However, the factor of operation constraints limits our choice of machine learning algorithms. In particular, since we can only submit 20K predicted tickets to ATDS every week, we desire a method that maximize the number of true predictions within these 20K.

In the following, we first formally define the ticket prediction problem, and then discuss in detail how we address these two factors while designing the ticket predictor.

4.1 Problem Definition

Let $\mathcal{U}$ be the set of all DSL customers (or the set of all DSL lines). For each DSL line $u \in \mathcal{U}$, let $\mathbf{l}^{(u,t)} := \{l_i^{(u,t)}; 0 \leq i \leq 25\}$ be the line measurements corresponding to the DSL line u at the time t (we drop the superscripts when the context is clear). Let $NT(u,t)$ be the length of the time period from t to the time of the immediate next customer ticket observed registered by u. Let $Tkt(u,t,T) := I(NT(u,t) < T)$, where I is the indicator function and $Tkt(u) = 1$ if a ticket from u is observed between $[t, t+T]$ and $Tkt(u) = 0$ otherwise (note we replace $Tkt(u,t,T)$ with $Tkt(u)$ when t and T are predefined). The ticket predictor outputs the probability that a ticket occurs within an interval T after time t, and can be expressed as a function f:

$$P(Tkt(u)|\mathbf{h}) = f(\mathbf{h}) \tag{1}$$

where the history $\mathbf{h} = \{\mathbf{l}^{(h,t_k)}), 1 \leq k \leq K\}$, K is the length of the history, and $t_K < t$ represents the time of the most recent line measurements. A shorter time interval T implies that the corresponding customer edge problems lead to immediate customer tickets (e.g., the problems that often cut off the connection); in contrast, a longer T value allows problems that are less noticeable by customers, such as intermittent connections, slow speed, etc. In addition, a longer time interval may also imply that a customer is not on site when the problem occurs, and the problem is only reported when the customer comes back home. Therefore, in our experiment, we try to address all these situations by choosing a longer T equals to 4 weeks.

4.2 Encoding Line Measurement History

We now discuss how to encode the time-series of line

[1]Customer anonymity was preserved during this study by hashing each customer phone number to a unique anonymous identifier prior to joining these datasets.

measurements into line features so that they can be readily used by the ticket predictor for detecting potential customer edge problems. Our basic idea is to define various types of features to reflect different properties of the line conditions embodied in these time-series. We define three types of features: basic features, delta features, and time-series features (Table 3, row 1-3). In practice, we expect the ticket prediction to be conducted every Saturday together with the weekly line measurement. Therefore, the basic features represent the current conditions of individual DSL lines every Saturday. The delta features record the change in the basic features compared with the feature values in the previous week, which measures the change with respect to the short term history. In comparison, the time-series features measure the deviation of the basic features in comparison to a long term measurement history.

Type	Name	Def.
History	Basic	l_i^K
	Delta	$l_i^K - l_i^{K-1}$
	Time-series	$(l_i^K - \bar{l}_i)/var(l_i)$
Customer	Profile	$l_i^K/profile(l_i)$
	Ticket	Time from the most recent trouble ticket
	Modem	Percentage of times that the modem is off
Derived	Quadratic	$(l_i^t)^2$
	Product	$l_i^t \cdot l_j^t$

Table 3: DSL line measurement features.

In addition to these features regarding line conditions, the chance of a potential trouble ticket also depends on certain characteristics of the customers. For example, a downloading speed of 128kbps on a slow-speed DSL line indicates a very good line performance. However, such a speed often leads to a trouble ticket on a high-speed line. We therefore introduce three kinds of customer related features in Table 3 (row 4-6) to address this issue. The profile features are defined as the basic features divided by the expected feature values specified by the corresponding customer profile, e.g., the minimum downloading bit rate or the maximum downloading bit rate, etc. The ticket feature records the time from the most recent trouble ticket to t_K. Its purpose is to address the correlation between tickets from the same customer. For example, if there has been a ticket in the past three days and problem is not fixed, the likelihood of repeat tickets will increase. The modem feature represents the percentage of times that the DSL modem is off during the test, which reflects the usage pattern of a customer. When a modem is off during the test, we have a missing record for that customer.

The third type of features are derived features (Table 3, row 7-8), which are computed from history features and customer features[2]. The quadratic features model the variance of each variable and the product features address the interactions between pairs of variables[3].

[2]We convert each categorical variables with m values into m binary variables so that the derived features are defined on all history features and customer features.

[3]The derived features are introduced to compensate for the fact that chosen *BStump* algorithm ignores the interactions among features.

4.3 Top-N Average Precision Feature Selection

Our next step is to use machine learning algorithms to extract the correlation between these line features and future customer tickets. Due to the 20K capacity constraint from ATDS, we want to choose a machine learning algorithm that maximize the accuracy of the top 20K predictions. However, most of the state-of-the-art algorithms focusing on ranking accuracy are usually computationally expensive and hence cannot be applied for ranking millions of DSL lines. Our solution to this problem is to use a scalable machine learning algorithm that maximize the overall accuracy. To deviate the "attention" of such an algorithm onto the accuracy of the top 20K predictions instead of the overall accuracy, we propose a novel top-N average precision metric to select only features that provide better accuracy on the top 20K predictions. We elaborate the feature selection method below.

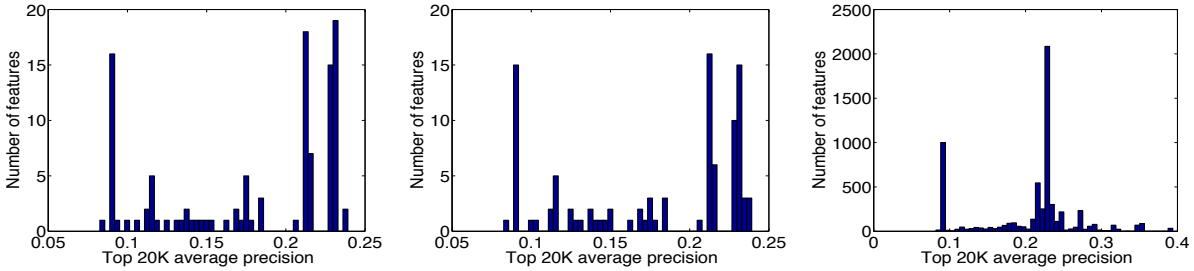
We first define the top-N average precision. Let N be the total number of top predictions selected. $Prec(r) := 1/r \sum_{i=1}^{r} Tkt(u_i)$ is the precision computed based on the first r predictions. The top-N average precision is defined as $AP(N) := \sum_{r=1}^{N} Prec(r) \times Tkt(u_r)/N$. Hence $AP(N)$ measures the sum of precisions for all true predictions averaged by N, and $AP(N)$ favors ranking the lines with future tickets higher than the lines without future tickets.

We apply a greedy feature selection method based on $AP(N)$ as follows. We first construct a ticket predictor (see Section 4.4 for description of the predictor) given each individual feature on a training dataset, and test the predictor on a separate test set. We then compute $AP(N)$ for each individual feature based on the result on the test set. The features associated with the highest $AP(N)$ values are selected. We demonstrate the histogram of $AP(20K)$ for all the features in Fig. 4. We observe strong bimodal shapes in Fig. 4[a] and Fig. 4[b] for history and customer features and the associated the quadratic features, respectively. We choose 0.2 as our threshold to select the features which are significantly more accurate than the other features of the same type. Because each product feature is formed by combining two other features, we hence expect the $AP(20K)$ for the product feature to be significantly higher than any of the two features. Therefore, we choose a higher threshold for selecting product features (greater than 0.3).

4.4 Learning the Ticket Predictor

Let $\mathbf{x}$ be the set of features selected from the original history of line measurements $\mathbf{h}$. Constructing the ticket predictor is equivalent to solving the regression problem $P(Tkt(u)|\mathbf{x}) = f(\mathbf{x})$. We use existing customer tickets as the ground truth and apply supervised machine learning techniques to learn f automatically. To train f, we treat the DSL lines with future tickets within 4 weeks as positive examples and the rest of the lines as negative examples.

Due to the fact that some customer edge problems may not be reported by customers, training data corresponding



(a) History and customer features (b) Quadratic features (c) Product features

Figure 4: Top 20K average precision for different types of derived features

to these problems are mislabeled as negative examples other than positive examples. Because of the existence of such noise in the training data, sophisticated non-linear models overfit easily, we hence choose a linear model for f which is more robust against noise. The *Adaboost* algorithm with decision stumps (i.e. one-level decision trees) [16], which we refer to as *BStump* in this paper, is selected to construct the model. The linear model from *BStump* was shown to be the most scalable while having an accuracy comparable to sophisticated non-linear classifiers [7].

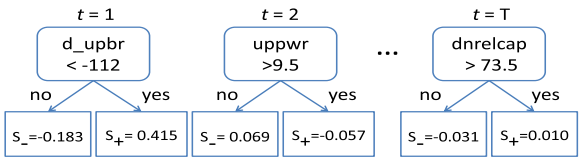


Figure 5: The schematic view of a *BStump* classifier.

Fig. 5 gives a schematic view of *BStump*. During training, we specify the number of iterations T (or the number of the weak learners) used by the algorithm. At iteration t, the algorithm selects one particular line feature and the corresponding feature value δ that best partitions the training data, weighted based on the classification result in iteration $t-1$, into positive and negative examples. The algorithm creates a decision stump using the selected feature as a weak learner g_t. Each weak learner outputs S_- (less likely to have a future ticket) for a feature value below δ (for continuous features) or not equal to δ (for categorical features), and outputs S_+ otherwise (more likely to have a future ticket). For example, in Fig. 5, the first weak learner ($t = 1$) tests the delta uploading bit rate against $\delta = -112$, if the uploading rate in the previous week is more than 112kbps above that in this week, the weak learner outputs a score $S_+ = 0.415$, and $S_- = -0.183$ otherwise. A total score corresponding to a combination of the outputs from all weak learners is computed and a threshold is applied to compute a binary outcome. The data weights are adjusted in order to best reproduce the ground truth on all line measurement records. This process iterates until T weak learners are generated.

At run time, for each line measurement record $\mathbf{x}$, T scores are generated by the weak learners, and these scores are summed up as the prediction $f(\mathbf{x}) := \sum_{t=1}^{T} g_t(\mathbf{x})$. The score f is then converted to the posterior probability $P(Tkt(u)|\mathbf{x})$ using logistic calibration.

5. EXPERIMENTAL RESULTS

We use line measurement records from 08/01/09 to 09/31/09 as our training data, and the data in the four contiguous weeks starting from 10/31/09 as our test data. The line measurements from 01/01/09 to 07/31/09 are history records for computing time-series features and customer related features. We use Boostexter [16] as the *BStump* implementation. The number of iterations is set to 800 based on cross-validation[4].

5.1 Evaluation of Ticket Predictor

As the first step, we compare the proposed top-N average precision method (N=20K) with other well-known feature selection methods. Due to the computational complexity of certain feature selection methods and the large number of derived features, we only use history features for the comparison. In particular, for each feature selection method, the top 50 features are selected according to the criteria in Table 4, and a classifier is constructed using these 50 features.

Methods	Feature selection criteria
AUC	Maximum area under the ROC curve.
Average precision	Average precision value on *all* the samples.
PCA	Top principal components.
Gain ratio	The total entropy decrease of the result attribute by knowing one particular feature.

Table 4: Other feature selection methods.

We note that we use accuracy in this paper for evaluating the performance of the ticket predictor, which is defined as the proportion of subscribers associated with the top N predictions who have issued tickets within 4 weeks from the prediction time. However, such a metric is very conservative. An *incorrect prediction* may be related to a real DSL problem but somehow the customer did not issue a ticket for various reasons. We shall discuss these reasons in detail in Section 5.2.

[4]When the number of iteration is equal to 800, training the classifier on 1 million training records requires around 2 hours on a typical server machine without parallelization. Only less than 15 minutes are needed for ranking all the DSL lines (several million).

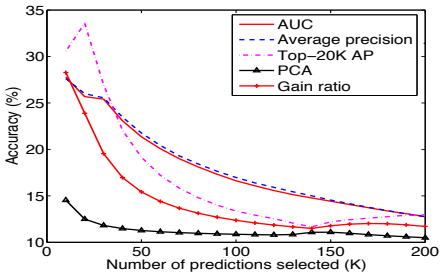


Figure 6: Feature selection methods.

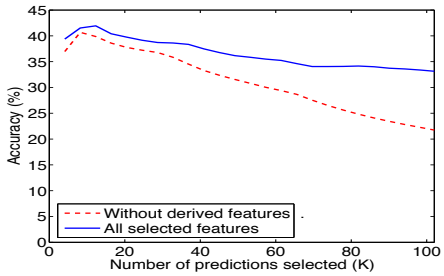


Figure 7: Ticket prediction results.

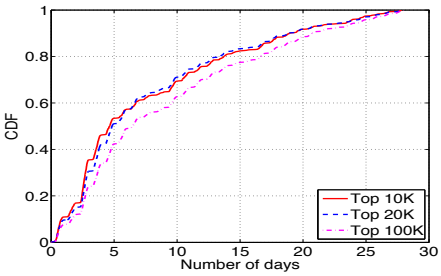


Figure 8: CDF of ticket coming time.

We illustrate the accuracy of different feature selection methods in Fig. 6, where the x-axis represents the number of top predictions selected and the y-axis stands for the accuracy of these top x predictions. We observe that the proposed top-N AP method outperforms all the other methods significantly when less than 20K predictions are chosen. However, as more predictions are selected, the accuracy from the top-N AP method becomes lower than the popular AUC based method. This phenomenon implies that, even though the boosting algorithm focuses on maximizing the overall accuracy, the features chosen by the top-N AP method effectively select more true predictions to be within the top 20K. In this way, we can make better use of the extra ticket handling capacity provided by the ATDS system to proactively resolve more potential customer edge problems. We note that in our top-N AP method, N is a tunable parameter, which can be enlarged when more predictions can be accommodated by ATDS.

We next study the performance of the ticket predictor on different line measurement features in Table 3. Fig. 7 shows the prediction results under different feature sets, where the x-axis stands for the number of predictions selected and the y-axis represents the accuracy or precision in the top x predictions. The dotted curve corresponds to the accuracy when only history and customer features are used, while the solid curve represents the accuracy when all the features in Table 3 are used. The top-$20K$ AP method is applied to select features from both feature sets.

From Fig. 7, we achieve good accuracy (37.8%) with history and customer features for the top 20K predictions. The addition of derived features further boosts the accuracy to 40%. More specifically, when N reaches the maximum capacity of 20K, we achieve 2 true predictions for 3 incorrect predictions (or a ratio of 1:1.5)[5]. This means that for more than 8K of the predicted lines, we observe at least one ticket within four weeks after our prediction. This demonstrates the effectiveness of incorporating the covariance of features into the classifier by encoding them explicitly using derived quadratic features and product features.

5.2 Discussion of the Prediction Results

[5]We have also compared the results using features from AUC and gain ratio methods, which again are worse much than the top-N AP method for the top 20K predictions. We omit the detail here due to space limit.

From the operation perspective, the effectiveness of a proactive solution is affected by two factors: 1) the number of actual customer edge problems within the 20K predictions; 2) the time allowed for resolving predicted problems before customers complain.

From the evaluation result in Section 5.1, we achieve an accuracy around 40% for the top 20K predictions, i.e., around 12K incorrect predictions. Note that in such an offline evaluation, we rely on actual customer tickets to evaluate the accuracy of the proposed ticket predictor, i.e., a ticket is considered to be correctly predicted if and only if a ticket associated with a customer edge problem is observed within one month after the prediction. We use one month here to give the customers enough time to report the problems dealing with intermittent connections or slow browsing problems as they are hard to perceive or more tolerable. Still, the reported problems from the customer tickets only comprise a subset of all the customer edge problems. In the following, we explore two scenarios when a real problem will not be reported by the customer.

Outage problems and IVR. The first scenarios is related to the outage problems. When an outage problem is detected at a certain area (e.g., from customer tickets or system alarms), an interactive voice response (IVR) system will be set up to automatically inform customers who call from the same area about the outage problem. In this case, the customer does report a problem (caused by the outage) but no ticket is issued since the phone call is handled by IVR instead of customer agents. We explore this scenario as follows. For each incorrect prediction, we first identify the corresponding DSLAM that it is associated with. We assume the incorrect prediction belongs to the IVR scenario if at least one outage problem is observed in the tickets from the customers belonging to the same DSLAM within T weeks from the prediction time.

	1 week	2 weeks	3 weeks	4 weeks
% of incorrect predictions	12.7	18.4	26.4	31.5
Coef. for outage prediction	0.0733	0.0752	0.0787	0.0784
P-value	0.0021	0.0013	0.0027	0.0011

Table 5: Incorrect predictions explained by outages.

By varying T from 1 week to 4 weeks, we show the proportion of incorrect predictions that can be explained by the IVR scenario in Table 5 (row 1). Around 12.7% incorrect

predictions are likely the artifact of IVR for outage problems when the outage problems are observed within a week from the prediction time. This proportion increases to 31.5% when at least one outage problem is reported in 4 weeks.

In fact, we have observed a correlation between predictions and future outage problems. Let $outage(d, t, T)$ be a binary event, where $outage(d, t, T) = 1$ if at least one outage problem occurs in DSLAM d between $[t, t + T]$. We quantify the correlation between ticket predictions (in the top 20K) and future outage problems using logistic regression: $logit(\#predictions) \sim outage(d, t, T)$. We vary T from 1 week to 4 weeks. The coefficients and P-values from the regression results are presented in Table 5 (row 2-3). We observe that there is always a strong (P-values less than 5%) positive (coefficients greater than 0) correlation between the future outage problems and number of predicted customer edge problems from a certain DSLAM. Based on this observation, the number of predictions associated with a DSLAM can be used as an indicator for future outage problems. In addition, we can group predictions by DSLAMs and send one truck to resolve most of the problems in a given DSLAM.

Customers not on site. In the second scenario, a customer may not use the DSL service when the predicted problem occurs, thus she will not notice the problem and no customer ticket is issued. To identify incorrect predictions corresponding to this scenario, we collect daily aggregated byte information for individual customers under two BRAS servers. We consider a customer to be not on site when no traffic is observed from that customer from one week before the prediction time until one week after the prediction time. For 12K incorrect predictions, we identify 108 subscribers with corresponding byte information. However, 18 out of these 108 subscribers (16.7%) are classified as not on site. This means such 16.7% predictions are plausibly with real customer edge problems, but the customers do not realize the problems since they are away. This also provide us a way to prioritize the 20K predictions by fixing lines with more customer activities first. Even though we only have a small sample of all the DSL customers, we expect the same observation holds for the DSL customers connecting to other BRAS servers. In addition, there might be customers belonging to both scenarios that we have discussed so far. We will investigate this in the coming trial of NEVERMIND.

Determining the urgency to solve a problem. We next study the time interval from the time of prediction of a problem to the time when a customer reports the problem. The purpose is to understand how much time we have from when a prediction is made before the customer starts complaining. The more time we have, the more chance for resolving the problem and the higher chance for reducing future tickets. We illustrate the distributions of the time intervals for the top 10K, top 20K and top 100K predictions in Fig. 8. We observe in Fig. 8 that around 80% of all the predicted tickets come within two weeks. This implies higher accuracy for predicting tickets in the near future, which is not surprising, since the correlation between line measurements and future customer tickets becomes weak as the time gap increases.

We consider that we miss a ticket if we fail to resolve a predicted ticket before the customer complains. From Fig. 8, if we fix all the predicted problems by Monday (allowing two days for fixing the problems), we will miss at most 15% of the tickets. Only at most 20% of the tickets are missed if we fix all predicted problems in three days. We thus have plenty of time to handle the predicted customer tickets.

6. LOCATING PROBLEMS

In this section, we briefly introduce the design and evaluation of the trouble locator. An information-theoretic justification of the proposed inference model can be found in the technical report version of this paper [8].

6.1 A Simple Experience Model

When a dispatch is scheduled, a field technician typically goes to a number of locations, e.g., DSLAM or customer home network, and test multiple devices until the problem is located. To save time, the technician needs to prioritize all the possible locations in the order of decreasing likelihood of being the problem location. Without additional information, the best ranked list is based on the prior probability that problems occur at a given location in the past.

This simple experience model can be improved in several ways. First, technicians can be provided with a better ranked list of locations using existing line measurement data introduced in Sec. 3. The key problem here is to infer problem locations from line features. Second, the time spent testing one location may differ significantly from the time spent testing another, and, if these locations have equal prior probabilities of being the cause of failures, a technician will save time by starting with the one which is the fastest to test. Third, technicians want to minimize the time they spends moving from one location to another.

This paper will focus on what we can do with the available information, namely, the first improvement where the location is inferred from line features. A this point, the time/cost for testing a location, and the time/cost for moving from one location to another are not available and considered as constants. In the following, we introduce the inference model employed by the trouble locator.

6.2 Flat and Combined Inference Models

The direct inference model, which we refer to as the *flat* model, ranks all the locations according to the probability of being the root cause of a customer edge problem. Let $\mathbf{x} \in \mathcal{R}^n$ denote all the line features in Table 3. Let $C_{i\cdot}, 1 \leq i \leq 4$ be one of the four major problem locations, e.g., the home network (HN), and C_{ij} be a specific disposition at $C_{i\cdot}$, e.g., the DSL modem at HN (see Fig. 2).

The flat approach consists of training a one-versus-other binary classifier $f_{C_{ij}}$ for each of the C_{ij} dispositions (the negative classes corresponds to all examples that do not be-

long to C_{ij}). The output from the classifier is then converted to $P_{ij}(C_{ij}|\mathbf{x})$ through logistic calibration, which represents, given the current line measurements $\mathbf{x}$, the likelihood that the problem is caused by the location associated with the disposition C_{ij}. Hence all the locations are ranked according to this probability and the technician will follow such a rank list during the dispatch.

To enhance the accuracy of the flat model, we propose a combined model, which explicitly takes into consideration the hierarchical disposition of the classes in Table 1. The basic idea is to combine two flat models, one for the disposition and one for its parent major location, together for the inference. In particular, for each disposition C_{ij}, we learn two classifiers $f_{C_{i.}}$'s and $f_{C_{ij}}$'s and combine outputs from these classifiers as the final inference result $P^{adj}_{ij}(C_{ij}|\mathbf{x})$ through logistic regression as follows (γ's are the coefficients from the logistic regression):

$$P^{adj}_{ij}(C_{ij}|\mathbf{x}) = \frac{1}{1+exp(-\gamma^1_{ij} f_{C_{ij}}(\mathbf{x}) - \gamma^2_{ij} f_{C_{i.}}(\mathbf{x}) - \gamma^0_{ij})} \quad (2)$$

6.3 Evaluating the Trouble Locator

We select 7 weeks of line measurement data from 08/01/09 to 09/18/09 for training and another 7 weeks data from 09/19/09 to 11/06/09 for testing. Based on the disposition notes we select 52 dispositions (C_{ij}'s,$1 \leq j \leq 52$) that appear more than 20 times in the data, which account for 81.9% of all the customer edge problems. We also categorize these 52 dispositions into one of the four major locations ($C_{i.}$'s,$1 \leq i \leq 4$) in Table. 1. For each disposition (C_{ij}) or major location ($C_{i.}$), we train a flat model ($f_{C_{ij}}(\mathbf{x})$ or $f_{C_{i.}}(\mathbf{x})$) using the *BStump* algorithm. We then assemble the inference model of each C_{ij} with the model of its parent major location $C_{i.}$ to form a combined inference model using Eq.2. Using these 52 combined models learned by the *BStump* algorithm, a ranked list of dispositions can be created based on the posterior probabilities $P(C^{adj}_{ij}|\mathbf{x})$.

The number of iterations for *BStump* is set to 200 based on cross-validation tests. We use all the line features presented in Table 3 to build the model.

Illustration of the combined inference model. We demonstrate the combined inference model in Fig. 9, which infers the likelihood that, given the measurement data ($\mathbf{x}$), the problem is caused by the inside wiring (`IW`) in the home network (`HN`).

The nodes at the bottom represent partitions of the values of different line features. The nodes in the middle represent the intermediate classifier scores of $f_{IW}(\mathbf{x})$ and $f_{HN}(\mathbf{x})$ using the *BStump* models. The top node is the final prediction result $P(IW^{adj}|\mathbf{x})$. The scores on the arrows from the bottom nodes to the intermediate nodes are extracted directly from the weak learners in the *BStump* algorithm, which indicate how strong the intermediate nodes depend on the feature values in the bottom nodes. A positive/negative score means the chance that the corresponding problem is caused by the intermediate node will increase/decrease if the feature value falls with the range specified by the bottom node. After $f_{IW}(\mathbf{x})$ and $f_{HN}(\mathbf{x})$ are obtained, they are further combined into $P(IW^{adj}|\mathbf{x})$ using Eq. 2.

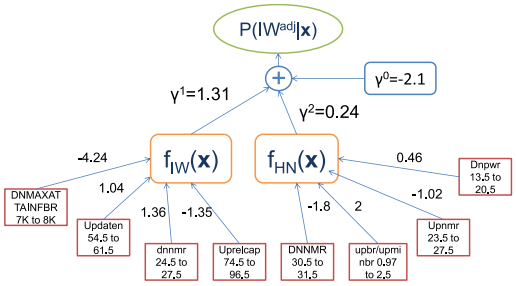


Figure 9: Combined model for the `IW` problem at `HN`.

Accuracy of the trouble locator. In general, the ranks of locations from the trouble locator using both types of models are much more accurate than the *basic ranks*, based on the simple experience model (Section 6.1). For example, using the basic ranks, in order to locate 50% of the problems, a maximum of 9 tests are needed. In comparison, using either the flat model or the combined model, only a maximum of 4 tests are required. This means that, in half of the dispatches, a technician can save more than 50% of the time using the trouble locator.

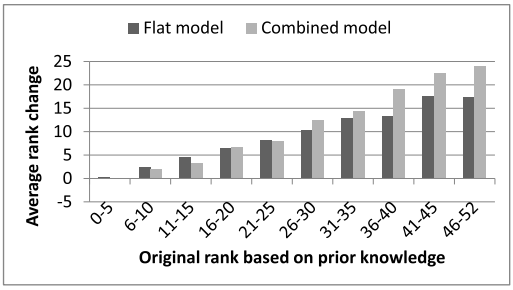


Figure 10: Rank change caused by the trouble locator

In Fig. 10, we illustrate the improvement over the basic ranks for different problem locations. We bin all the customer tickets in the test data according to their basic ranks, and then compute in each bin the average rank change. We observe that both types of inference models can help improve basic ranks significantly. For example, for the problems with basic ranks between 16 and 20, both models can boost the average rank by 4. In other words, the technician can save time for testing 4 more devices during each dispatch on average. Such improvement becomes more significant when the basic rank is deeper down the list. Comparing the performance of the flat model with the combined model, we see that the combined model further improves the ranks for the problems with lower ranks.

7. RELATED WORK

Troubleshooting is a key network management problem

which has received considerable attention in the literature. The most widely used troubleshooting solution in today's large commercial network is based on expert rules [1,2,11]. Summarizing expert rules relies heavily on domain knowledge and thus is expensive and does not generalize well.

To overcome the problem of using manual rules, many research works apply machine learning techniques to automatically extract inference rules using a pre-labeled training set, such as [3,5,6,14,17–21]. The basic idea in these works is to train a classifier which differentiates high level performance states based on network system-level metrics. Our problem differs in the special operational constraints in troubleshooting DSL problems, such as limited operation resources.

Another popular approach is to explicitly model the relationship among different network components as inference graphs and then compute their dependencies. Such dependencies can be measured at various levels of granularity. For example, Sherlock [4] models the dependency among network devices to detect high level problems like connectivity problems. In contrast, NetMedic [10] models the dependency of fine-grained components like processes to troubleshoot application failures. In addition, specific domain knowledge can also be used for constructing inference graphs, such as SCORE [12] and Shrink [9]. NetworkMD [15] automatically learn such inference graphs by correlating failure history using nonnegative matrix factorization. A limitation in these approaches is that they require access to measurements of individual components, which are too expensive to obtain in DSL networks.

In terms of the scale of the target network, one related work is GIZA [13] which focuses on troubleshooting performance issues in large IPTV networks. GIZA correlates heavy hitter events to locate problems on the hierarchical structure of the IPTV network, which affect multiple users at the closeby area. In comparison, our method targets primarily troubleshooting customer edge problems. Another key difference is that GIZA is a reactive approach which uses customer tickets as one of the data feeds; while our approach reduces future customer tickets.

8. CONCLUSION

In this paper, we presented NEVERMIND, a proactive approach for detecting and troubleshooting DSL customer edge problems. NEVERMIND utilizes existing sparse measurements of physical DSL line conditions to predict future customer tickets and solves them in advance to eliminate these tickets and potentially reduce customer churn. During the design of NEVERMIND, we accounted for the operational constraints and special properties of the DSL network, and proposed novel techniques, such as a top-N average precision based feature selection method and a combined inference model which incorporate the hierarchical structure of DSL networks, to enhance the accuracy of the approach. Evaluations using a whole year worth of data gathered from a large DSL network demonstrate that NEVERMIND was able to accurately predict thousands of tickets every week with high accuracy and significantly reduce the troubleshooting time. We are currently focusing on trialing an operational deployment in a large DSL network.

Acknowledgement

We would like to thank our colleagues from AT&T Customer Care for helpful discussions. The work is supported in part by the NSF grants CNS-0905037 and CNS-1017647, and the DTRA Grant HDTRA1-09-1-0050. Part of the work was done during an internship by the first author at ATT Labs – Research.

9. REFERENCES

[1] Gteko, inc. http://www.gteko.com.
[2] Open view, hp technologies inc. http://www.openview.hp.com.
[3] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker. Netprints: diagnosing home network misconfigurations using shared knowledge. In *NSDI'09*, 2009.
[4] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *SIGCOMM'07*, 2007.
[5] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, O. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *ICDSN'02*, 2002.
[6] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase. Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *OSDI'04*, 2004.
[7] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang. A modular machine learning system for flow-level traffic classification in large networks. Technical report, AT&T research, 2009.
[8] Y. Jin, N. Duffield, A. Gerber, P. Haffner, S. Sen, and Z.-L. Zhang. Proactively detecting and troubleshooting customer dsl problems. Technical report, AT&T research, 2009.
[9] S. Kandula, D. Katabi, and J.-P. Vasseur. Shrink: a tool for failure diagnosis in ip networks. In *MineNet'05*, 2005.
[10] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl. Detailed diagnosis in enterprise networks. In *SIGCOMM'09*, 2009.
[11] G. Khanna, M. Cheng, P. Varadharajan, S. Bagchi, M. Correia, and P. Veríssimo. Automated rule-based diagnosis through a distributed monitor system. *IEEE Trans. Dependable Secur. Comput.*, 2007.
[12] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren. Ip fault localization via risk modeling. In *NSDI'05*, 2005.
[13] A. Mahimkar, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and Q. Zhao. Towards automated performance diagnosis in a large iptv network. In *SIGCOMM'09*, pages 231–242, 2009.
[14] Y. Mao. Automated computer system diagnosis by machine learning approaches. Technical report, UPenn CIS Dept., 2005.
[15] Y. Mao, H. Jamjoom, S. Tao, and J. Smith. Networkmd: topology inference and failure diagnosis in the last mile. In *IMC '07*, 2007.
[16] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Mach. Learn.*, 2000.
[17] Y.-Y. Su, M. Attariyan, and J. Flinn. Autobash: improving configuration management with operating system causality analysis. In *SOSP'07*, 2007.
[18] H. J. Wang, J. C. Platt, Y. Chen, R. Zhang, and Y.-M. Wang. Automatic misconfiguration troubleshooting with peerpressure. In *OSDI'04*, 2004.
[19] Y.-M. Wang, C. Verbowski, J. Dunagan, Y. Chen, H. J. Wang, C. Yuan, and Z. Zhang. Strider: A black-box, state-based approach to change and configuration management and support. In *LISA'03*, 2003.
[20] C. Yuan, N. Lao, J.-R. Wen, J. Li, Z. Zhang, Y.-M. Wang, and W.-Y. Ma. Automated known problem diagnosis with event traces. In *EuroSys'06*, 2006.
[21] A. X. Zheng, J. Lloyd, and E. Brewer. Failure diagnosis using decision trees. In *ICAC'04*, 2004.