A MACHINE LEARNING APPROACH TO PROBLEMS IN COMPUTER NETWORK PERFORMANCE ANALYSIS

by

Mariyam Mirza

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2012

Date of final oral examination: 05/25/12

To Baba.

ACKNOWLEDGMENTS

I have worked with a number of faculty, staff and fellow students during my time as a graduate student. Thanks to Joel Sommers for spending a great deal of time mentoring me during my first couple of years in graduate school. Thanks to Jerry Zhu for teaching me about machine learning, and to Suman Banerjee for teaching me about wireless networking. Thanks to Michael Blodgett for helping me set up my experiments.

I faced some serious personal challenges in the second half of my time in graduate school. I fell ill with chronic fatigue syndrome and did not get a correct diagnosis for three years, which eventually required me to take time off from school. A couple of months after I recovered and returned to school, both my parents fell critically ill at the same time. I would not have been able to finish my degree had it not been for the support of a large number of people who helped me through these two crises.

My academic effectiveness dropped significantly during my illness. A big thank you to Paul Barford, my advisor, for sticking with me through it all.

Shanta Hattikudur and Jenni Petersen, thank you for being amazing friends in general and taking care of me during my illness in particular. You drove me to doctor appointments, took me to the grocery store and cleaned my apartment on a regular basis while being busy with your own Ph.Ds.

Basma Khan, Sabeen Jamil, Mrs. Mitra Irani, Nazia Khan, Dr. Akhtar Aziz Khan and the staff at Indus Hospital, thank you for taking care of my father over the last six months. Without your help, I would have had to leave school and return to Pakistan to look after Baba. Emma Seppala and YES+ Bucky, thank you for teaching me that inner peace is achievable, and worth fighting for, regardless of the severity of the external turmoil.

Baba, thanks for being the best father a girl could ever have. You have been my biggest champion, always believing that I can overcome seemingly impossible challenges to accomplish everything I want. Many talk of unconditional love. You have truly walked the walk in the last six months, choosing to fight for your life without any immediate family by your side, so that my life and my education are disturbed as little as possible by your illness.

Last but not least, thanks to all the parrots – Tingu, Bingu, Pingu, Chingu, Beerbal, Bella, Neville, Pumpkin and Houdini – who have owned me during my time in graduate school. You are the best study-buddies, and having you by my side made the long hours and all-nighters significantly less stressful. When I was too discouraged to fight for my health for my own sake, I fought for it so you would not have to go back to the shelter. Thanks for all the love, joy and companionship you bring to my life.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

DISCARD THIS PAGE

LIST OF TABLES

DISCARD THIS PAGE

LIST OF FIGURES

Figure Page

DISCARD THIS PAGE

NOMENCLATURE

ACK	Acknowledgement
AP	Access Point
BER	Bit Error Rate
CBR	Constant Bit Rate
CTS	Clear To Send
DCF	Distributed Coordination Function
EWMA	Exponentially-Weighted Moving Average
GPS	Global Positioning System
GSR	Gigabit Switch Router
IP	Internet Protocol
HAL	Hardware Abstraction Layer
HTTP	Hypertext Transfer Protocol
LAN	Local Area Network
MAC	Medium Access Control
MIB	Management Information Base
MIMO	Multiple Input Multiple Output
OC	Optical Carrier
OFDM	Orthogonal Frequency Domain Multiplexing

PHY	Physical Layer
RAM	Random Access Memory
RBF	Radial Basis Function
RF	Radio Frequency
RON	Resilient Overlay Network
RSSI	Received Signal Strength Indicator
RTS	Request To Send
RTT	Round-Trip Time
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machines
SVR	Support Vector Regression
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAIL	Wisconsin Advanced Internet Laboratory

A MACHINE LEARNING APPROACH TO PROBLEMS IN COMPUTER NETWORK PERFORMANCE ANALYSIS

Mariyam Mirza

Under the supervision of Associate Professor Paul Barford
At the University of Wisconsin-Madison

Computer networks are becoming increasingly complex with time due to the introduction of sophisticated technologies and protocols and the increase in network size in terms of users, volume of data transferred, and available bandwidth. This increase in complexity means that in many cases existing network performance analysis methods are no longer effective, and that there is a need for more powerful methods.

In this dissertation, we propose the use of Support Vector Machines (SVMs), a powerful state-of-the-art machine learning technique, as a network performance analysis framework that can cope with increasing network complexity. Support Vector Machines are a suitable candidate framework in this situation because of their ability to handle complicated non-linear relationships between large numbers of variables while retaining high computational efficiency.

To make a case for the effectiveness of SVMs as a framework for network performance analysis, we devise SVM-based solutions for three network performance analysis problems. The three problems that we consider present very different challenges, but they have in common the property of becoming more difficult to solve as network complexity increases. Finding effective solutions to a set of problems that present different challenges is important for establishing SVMs as a useful general-purpose framework rather than an ad hoc solution for specific situations. We find that in each case SVM-based solutions improve on existing solutions significantly in areas such as accuracy, measurement overhead and ease of practical deployment. Hence we conclude that SVMs provide an effective framework for network performance analysis in the face of growing network complexity.

Paul Barford

Chapter 1

Introduction

1.1 Motivation

The complexity of computer networks continues to increase for many reasons. One is the introduction of new technologies, protocols and optimizations at all network layers to improve network throughput, reliability and security. Another is the presence of many different flavors of a given protocol or optimization. A third is the increase in network size in terms of the number of users and the volume of data transferred. A fourth is the increase in range of both available bandwidth and delays.

This increase in network complexity makes the task of analyzing network performance more difficult. However, network performance analysis continues to be important. The design of improved network technologies is often driven by an understanding of the performance deficiencies in existing ones. Network management tasks such as provisioning, tuning and debugging depend on an understanding of network performance.

Each of the reasons for increase in network complexity listed above lead to specific challenges for network performance analysis. The most important criterion on which any attempt at network performance analysis is evaluated is accuracy. However, as discussed below, network complexity forces analysis efforts to strike a balance between maximizing accuracy and completely overcoming all these challenges.

The presence of a large number of technologies, protocols and optimizations at all network layers increases the number of variables that effect network performance. The identification of all variables that effect performance is challenging, as is capturing the exact relationship between the variables and a performance metric such as throughput. A consequence of this problem is that performance analysis efforts tackle simplified versions of protocols or systems rather than fully general production versions because it is too difficult to accurately analyze production systems. We refer to this as the *analysis comprehensiveness* challenge.

The presence of many different flavors of a given protocol or optimization means that even if the analysis comprehensiveness problem has been solved for a specific combination of protocols and optimizations, the analysis will need to be constantly updated as the protocols and optimizations are tweaked. We refer to this as the *extensibility and maintenance* challenge.

Increase in network size in terms of the number of users and volume of data, and increase in the range of available bandwidth and delay mean that network performance analysis techniques have to be able to *(a)* function effectively in a wide range of operating conditions across different networks, and *(b)* adapt quickly to changing network conditions in the same network. We refer to the first challenge as the *robustness* challenge, and to the second as the *agility* challenge.

Even if all of the above-mentioned problems are solved, it may be difficult in practice to acquire the information necessary to analyze network performance. This can occur due to system opaqueness, *e.g.*, in wide area paths the only available measurements are often from the perspective of the path end points, and the internal state of the network has to be inferred from these measurements. This can also occur in the form of measurement traffic overhead or because the number of measurements required for performance analysis increase with the number of nodes in the network. We refer to this as the *practicality* challenge.

In this dissertation, we focus on network performance analysis problems that involve either trace based classification of deployed protocols and algorithms or the prediction of network performance metric values. These two classes of problems become more difficult to solve as network complexity increases, so they provide good test cases for efforts focused on developing novel techniques to handle the problem of network performance analysis in the face of growing network complexity. Classification problems become more difficult because the increasing sophistication of protocols and algorithms means that they can can no longer be identified by short characteristic packet patterns. Prediction problems become more difficult because the number of factors affecting the target metric increases, and the relationship between the relevant factors and the target metric becomes more complex.

The types of protocol and algorithm classification problems we focus on in this dissertation are *in the dark* problems. These are problems for which classification is done based on implicit information contained in passively acquired packet traces rather than based on explicit queries or probes. Implicit information refers to any information that can be gleaned from passive packet traces, such as PHY, MAC and transport header fields, packet interarrival time distributions and whether a host is a consumer or producer of data.

Most of the prior efforts involving *in the dark* protocol and algorithm classification focus on identifying the classes of applications (web, peer-to-peer, mail, transactional) or application layer protocols (HTTP 1.0 versus HTTP 1.1, different peer-to-peer protocols) in use. There are two types of classification techniques, those which classify applications based on flow properties such as packet sizes and packet interarrival times [106, 85, 41, 89, 133, 35, 64, 21, 20, 111] and those which classify applications based on their *social* behavior, *i.e.*, by considering the number of different hosts they communicate with and the number of different ports they use [67, 11]. The above-mentioned techniques have high classification accuracy. These techniques meet most, if not all, of the analysis comprehensiveness, extensibility and maintenance, robustness, agility and practicality criteria desired of performance analysis techniques. The techniques meet the analysis comprehensiveness and robustness criteria because they exhibit high accuracy on real or realistic packet traces. Early classification techniques [21, 20, 111] are extremely agile because they can classify applications based on the first handful of packets in a flow. And *in the dark* classification is the only practical classification approach for production wireline and wireless networks because it does not assume any active cooperation from the clients in a network and works even when the data payload is encrypted.

All of the above-mentioned studies deal with application layer classification problems. All the flow-feature based studies demonstrate that a small number of fairly obvious features hold the key to highly accurate classification, and all the studies based on social behavior hinge on the presence of distinguishing communication patterns that are easy to identify and isolate. The classification problem we consider – identification of 802.11 rate adaptation algorithms – is a MAC-layer problem. As a result, transport and application layer features such as packet size, interarrival time and social behavior patterns, which are some of the most useful distinguishing features for application identification, are of little use. We need to find a novel classification technique because in our problem the algorithms to be classified can assume a much larger number of possible states than application protocols, and the distinguishing features are non-obvious.

Prediction-based studies predict future values of network performance metrics based on past values of the metric and/or the values of related end host and path properties. The performance metric predicted most often by such studies is network (TCP) throughput. Examples of throughput prediction studies for wireline environments include [96, 84, 87, 75, 98, 31, 10, 12, 128, 81, 120, 54], and for wireless environments include [23, 28, 27, 97, 105, 47, 68, 103, 92, 80].

There are two common types of prediction studies, *formula-based* and *history-based*. Formula-based studies predict future values of the metric of interest based on a mathematical formula that captures the relationship between network path properties (*e.g.,* loss and delay) and the metric of interest (*e.g.,* throughput). History-based studies predict future values of the metric of interest based on time-series analysis of past values of the metric. These two approaches make constrasting tradeoffs in coping with the challenges associated with system complexity.

Formula-based approaches deal with the analysis comprehensiveness challenge by modeling simplified versions of the system under study. It takes several incremental efforts to create full system or near full system models. An example from the wireline domain is the [84, 98, 31] sequence for TCP throughput modeling for wireline networks where [84] considers only losses due to triple duplicate ACKS, [98] adds analysis of losses due to timeouts, and [31] further completes the analysis by adding connection establishment and slow start latencies. An example from the wireless domain is the [47, 97, 105, 68, 103, 92, 80] set of throughput models where the simplest model [47] handles interference-free broadcast traffic only and the models become progressively more complete, with the most complete, [80], handling interference, RTS/CTS, multi-hop networks, and TCP traffic. Hence analysis realism can be an expensive and laborious undertaking for formula-based approaches. Maintenance and extensibility with formula-based approaches is challenging for the same reason, with new or significantly modified models required for improved or modified versions of protocols, *e.g.,* new performance models for TCP Vegas [108]. Formula-based approaches exhibit high agility, *i.e.,* respond quickly to changes in network conditions, assuming that up-to-date measurements of formula parameters are available. The robustness of formula-based approaches depends on analysis comprehensiveness. If the formula captures system behavior over a wide range of operating conditions, the approach is robust, otherwise it is not. The practicality challenges faced by formula-based methods are generally related to system opaqueness. Formula-based methods are designed based on assumptions regarding the measurements that will be available, and if these assumptions are incorrect, the method fails the practicality challenge. To completely handle the practicality challenge, formula-based methods have to construct multiple models based on measurement availability, *e.g.*, most wireline TCP throughput models assume that per-flow end-to-end RTT and loss rate is available, but [48] presents an alternate model that assumes that the only available information is via SNMP MIBs from routers at the granularity of aggregate traffic through the router. Constructing multiple models for the formula-based approach overcomes the practicality hurdle, but falls at the maintenance and extensibility hurdle.

History-based methods deal with the analysis comprehensiveness problem by side-stepping it. They are based on time-series analysis of the target metric, so assuming that accurate past measurements of the target metric are available, system complexity is not an issue. Consequently, history-based methods are able to handle real systems with significantly less effort than formula-based approaches. However, side-stepping the analysis comprehensiveness problem comes at a cost. Formula-based methods are able to provide insight into the factors that effect the target metric because the relationship is represented by the formula parameters, but history-based methods are unable to provide any such insight. Similarly, history-based methods handle maintenance and extensibility well with little extra effort, but with the same tradeoff with insight. History-based methods have poor agility because time-series analysis often includes smoothing to filter outliers, which results in slow response to genuine level shifts. History-based methods are quite robust as long as level shifts are not frequent. The practicality challenge faced by history-based methods is generally that of measurement overhead and delays, *e.g.*, in [54], the measurement samples required for TCP throughput prediction are bulk TCP transfers that introduce a large amount of measurement traffic into the network.

Our work is motivated by the desire to develop a network performance analysis framework that is able to provide robust performance analysis solutions in spite of the increasing complexity of computer networks.

1.2 Approach

As computer networks become increasingly complex, machine learning based methods have been applied to computer network analysis problems. These methods are particularly suited for computer network analysis because most network analysis problems involve understanding complex relationships between large numbers of variables, and learning-based methods are designed to handle just such problems.

In this dissertation, we introduce analysis based on Support Vector Machines (SVM), a powerful, state of the art machine learning technique, as a new and better design point in the spectrum of analysis methods for network performance problems involving classification and prediction. We are motivated by the need for analysis methods that can cope with increasingly complex and difficult to analyze computer networks.

To demonstrate that it is indeed a useful general purpose framework for network performance analysis rather than an ad hoc solution to narrow, one-off problems, SVM-based network performance analysis has to meet several criteria. First, it has to be able to provide effective performance analysis solutions for a diverse set of problems. Second, the solutions should be at least as accurate as, and ideally more accurate than, those based on existing methods. Third, the tradeoffs that SVM-based solutions make regarding the challenges of analysis comprehensiveness, maintenance and extensibility, robustness, agility and practicality should be no worse than, and ideally better than, those made by existing analysis frameworks.

In this dissertation, we apply SVM-based analysis to three network measurement problems, TCP throughput prediction for wireline networks, TCP throughput prediction for wireless networks, and identification of the rate adaptation algorithms deployed in a wireless network. These problems differ widely from each other, and effective solutions to these problems make a case for the feasibility of SVMs as a framework for network performance analysis.

We use SVMs because they are one of the best, if not the best, general purpose supervised learning methods available at present. They have a strong foundation in statistical learning theory [126] and also exhibit good performance for a wide variety of real world problems. SVMs do not conform to any particular parametric form, and due to their use of kernels (Chapter 3, Section 3.5) are able to capture complex non-linear relationships between large numbers of variables. SVMs are based on a convex optimization problem, so unlike methods such as neural nets and decision trees, they do not suffer from the problem of local minima. SVMs are computationally very efficient for several reasons. First, the theoretical basis of SVMs, that the best classification accuracy on test data is achieved by finding the correct balance between maximizing accuracy on training data and minimizing the complexity of the classifer, prevents SVMs from falling prey to the *curse of dimensionality*, and allows efficient solutions even for problems with high dimension attribute spaces (Chapter 3, Section 3.2). Second, the SVM optimization problem is solved in its more computationally efficient dual form rather than its primal form (Chapter 3, Section 3.3). Finally, the solution to the SVM optimization problem depends only on a small fraction of the input data (Chapter 3, Section 3.3), so SVMs are *sparse*. Chapter 3 presents a detailed overview of SVMs. In this section, we outline the challenges that have to be overcome to apply SVMs successfully to network performance analysis.

Thesis Statement

> As the complexity of computer networks increases, techniques that have been conventionally used for network performance classification and prediction, such as formula-based performance modeling and time series analysis, are no longer adequate in a number of scenarios, and more sophisticated techniques are needed. We claim that applying machine learning based methods to network performance classification and prediction results in significant improvements over conventional methods. This is due to learning based methods' ability to combine the

benefits, and eliminate many of the inherent tradeoffs, of conventional methods. We support our claim by finding learning-based solutions to three problems that present widely different performance analysis challenges. Our learning-based method of choice is Support Vector Machines (SVMs), a powerful state-of-the-art technique that has a strong foundation in statistical learning theory and is known to work well in practice. Our SVM-based solutions demonstrate learning-based methods' strengths over conventional methods in terms of the ability to handle complex real systems more easily, the ability to function accurately in a wide variety of network conditions and under rapidly changing conditions, and being low overhead and easy to deploy in practice.

The first step towards solving a problem using SVMs is to cast it into the SVM framework, in which a functional relationship is represented as a mapping between a *feature vector* and a *target value*. For a classification problem such as determining which of a set of 802.11 rate adaptation algorithms is deployed in a network, the feature vector consists of information such as the sequence of transmission rate changes, and the target value is the identity of the algorithm. For a regression problem such as TCP throughput prediction, the fields in the feature vector consist of path properties such as loss rate and RTT, and the target value is the TCP throughput. There is a *training* phase and a *test* phase. In the training phase, both feature vectors and target values are acquired from measurements of the network under consideration, and a predictor or classifier is constructed based on the relationship between the feature vectors and target values. In the *test* phase, only the feature vectors are available, and the predictor or classifier has to infer the target values.

The success of applying SVMs to a network performance analysis problem depends on two key elements, appropriate feature selection and effective training to construct the classifier or predictor.

Feature selection is the process of identifying the factors that are sufficient for the characterization of the target value over a wide range of operating conditions. The effectiveness of the features over a wide range of operating conditions is important because special cases of a problem can sometimes be solved by a small subset of features or even fallicious feature selection. Comprehensive feature selection is thus a manifestation of the analysis comprehensiveness challenge of network performance analysis. In some cases, *e.g.*, for TCP throughput prediction of wireline networks, the factors that effect the target value, throughput, such as loss rate and RTT, are well known and small in number. In other cases, *e.g.*, the identification of the 802.11 rate adaptation algorithms deployed in a network, features are non-obvious, and the number of features required for high accuracy is on the order of thousands.

Feature selection also has implications for the practicality of an SVM-based solution of a performance analysis problem. If the features are such that their values are easy to acquire without running into system opaqueness or measurement scalability issues, the SVM-based solution is more likely to be adopted in practice. An example is our solution for wireline TCP throughput prediction (Chapter 4, Section 4.3), where we show that while both RTT and available bandwidth are equally useful features in terms of prediction accuracy, the use of RTT is preferred because measuring available bandwidth is significantly more expensive in terms of the amount of measurement traffic introduced.

The robustness and agility of an SVM-based solution depends on the diversity of the training set used to construct the classifier or predictor, where diversity means that the samples in the training set demonstrate system behavior in as many different system states as possible. This suggests that if a very large and comprehensive training set is used, the classifier or predictor can be made infinitely robust and agile. Unfortunately, garnering such a comprehensive training set is neither always possible nor useful. The first issue is the amount of time required for gathering the training samples. For a scenario such as throughput prediction for wireless nodes moving at driving speeds, a quickly-generated if somewhat rough prediction is useful while an infinitely robust and agile one that takes a long time to generate is useless. The second issue is that it is not always possible to exercise all system states in training, *e.g.*, for TCP throughput prediction of wide area paths (Chapter 4, Section 4.4), the volume of cross traffic on the path cannot be controlled, limiting the comprehensiveness of the training set that can be gathered. Finally, even if the first two problems associated with creating comprehensive training sets are resolved, it is sometimes the case that training sets are not portable, *e.g.*, for wireless throughput prediction from one network to another, where the network environment is very complex and analysis has to be location-specific due to interference effects, node and building layout, etc. Hence problem-specific training protocols, with provisions for detecting errors and triggering re-training as necessary, have to be devised to make appropriate tradeoffs between comprehensiveness, practicality, robustness, and agility.

One of the greatest strengths of the SVM framework is that it allows for easy maintenance and extensibility. The need for maintenance and extensibility arises in cases where a problem instance somewhat different from one already solved occurs, such as throughput prediction for different TCP flavors or the addition of a new 802.11 rate adaptation algorithm to the set of algorithms to be identified. In such a scenario in the SVM framework, only the identification of new features that effect classification or prediction is required, and the SVM implicitly determines the relationship between the new feature set and the target value. In contrast, approaches that require explicit information to facilitate classification or prediction are very expensive to maintain, *e.g.*, every time a new 802.11 rate adaptation algorithm is developed, a model based on explicit enumeration of all possible algorithm states would be needed to enable its classification, and for formula-based throughput prediction new models have to be developed to predict throughput for different flavors of TCP. Hence the SVM-based approach supports maintenance and extensibility at significantly lower cost compared to approaches that require explicit modeling. And SVMs retain their computational efficiency even for high dimension feature vectors, so the computational cost of adding features is low as well.

The first problem for which we find an SVM-based solution is TCP throughput prediction for wireline paths. When multiple network paths exist between data senders and receivers, as is the case for content distribution, multi-homing, and overlay networks, there is a need to select the *best* path. The highest throughput path is generally considered the best path, and the capability to accurately predict TCP throughput can be used to select this path.

We investigate the relationship between bulk TCP transfer throughput and network path properties such as available bandwidth, queuing delay and packet loss in a controlled, high accuracy laboratory environment. **Our results show that, using lab-based passive measurements with perfect accuracy, SVM-based predictions are 3 times**

more accurate than history-based predictions, the most accurate predictor in prior work, with SVM-based predictions being within 10% of actual value 87% of the time for bulk transfers under heavy traffic conditions while history-based predictions are within 10% of actual value only 32% of the time. Since the lab-based perfect accuracy passive measurements are unavailable in the wide area, we evaluated SVM-based throughput prediction using less accurate but practical active measurements of network path properties. **Using these measurements, the accuracy of SVM-based predictions exceeds those of history-based predictions by a factor of 1.6, with 49% of SVM-based predictions being within 10% of actual value compared to 32% of history-based predictions.** We find that the above accuracy levels for both perfect lab-based passive measurements and practical active measurements can be achieved by using only queuing delay and packet loss rates for throughput prediction, and including available bandwidth provides almost no improvement in accuracy. This is good news, because measurements of packet loss and queuing delay are lightweight and introduce little measurement traffic into the network, while measurements of available bandwidth are considerably more heavyweight. **We find that the measurement traffic overhead of the SVM-based approach using active measurements of queuing delay and packet loss is a factor of 13 lower than that of a history-based approach.** Unlike prior work which focuses only on bulk transfers, our approach can predict throughput accurately for a wide range of file sizes. We found that a training set of only three file sizes can extrapolate and accurately predict throughput for a wide range of file sizes. Prior work [54] has shown that *level shifts* in path conditions pose difficulties for throughput prediction. We show that updating the predictor following level shifts allows our technique to adapt to level shifts almost immediately and to maintain high accuracy on paths where level shifts occur frequently. We complete our study by showing that our SVM-based predictor is robust not only in the laboratory environment, but also on a diverse set of 18 wide area paths.

The application of the SVM framework to TCP throughput prediction for wireline networks illustrates the power of the framework in many ways. The most important property of the SVM-based solution is its high accuracy even when throughput is variable, *i.e.*, when the average link utilization is 90% of the bottleneck. The fact that SVM does not conform to any particular parametric form and is able to capture complex relationships between variables enables robustness under challenging, variable throughput conditions. SVM allows for easy plug-and-play of candidate input features for throughput prediction, in contrast to formula-based predictors where explicitly deriving throughput using alternative path properties is an expensive undertaking. This capability enabled us to identify that lightweight RTT and loss measurements are sufficient to maintain high prediction accuracy and the more expensive available bandwidth measurements are unnecessary, making our solution low overhead and hence more desirable for practical deployments. The SVM-based approach can respond to level shifts immediately, *i.e.*, after observing a single training sample following a change in path conditions, making it significantly more agile than the history-based approach. This is because, due to its capability to consider as many input features as necessary, the SVM-based approach is able to distinguish between genuine level shifts and outliers much more effectively than the history-based approach. The flexibility and ease of feature inclusion allows SVMs to predict throughput for arbitrary-sized transfers, not just bulk transfers. The power of SVMs to capture complex relations between variables allows the predictor to extrapolate from three file sizes in the training set to a wide range of files sizes in the test set, allowing the SVM-based predictor to clear the analysis comprehensiveness hurdle and keep the training set size manageable at the same time.

The second problem to which we apply the SVM framework is TCP throughput prediction for wireless paths. Wireless applications can use throughput predictions to improve their performance in many ways. One use is the selection of the highest throughput AP by a wireless client when multiple APs are within range. Another is to determine the TCP friendly rate for non-TCP flows. Our work aims to design a TCP throughput mechanism that works not only for stationary clients in enterprise or hot-spot settings, but also for clients of *opportunistic networks*. Opportunistic networking is a new model of wireless connectivity in which guest users passing through the range of one or more open APs can gain temporary Internet access. Opportunistic networks can be stationary. However, over the last five years, many efforts have focused on understanding the limits of the opportunistic connectivity paradigm, developing protocols that allow efficient use of opportunistic connectivity, and developing applications for vehicular clients [94, 95, 29, 26, 58, 140, 52, 83, 40, 124, 123, 6]. Vehicular wireless clients present the most challenging scenario for throughput prediction for two reasons. First, the vehicular network environment is more dynamic and variable, *i.e.*, less predictable, than a stationary environment. Second, the vehicular environment imposes stringent limits on the amount of time available to generate a prediction. Studies such as [40] show that most vehicular clients spend 10 seconds or less within the range of a particular AP, so throughput predictions need to be generated in a small fraction of that time, no more than 1 second, to be useful to applications. Prior work on throughput modeling and prediction for wireless networks does not handle the stringent timeliness requirements of the vehicular environment. Our work aims to address this challenge.

TCP throughput prediction for wireless environments, whether they be stationary or vehicular, is extremely difficult, and differs significantly from TCP throughput prediction for wireline paths. The wireless network environment is much more variable at the physical (PHY) and medium access control (MAC) layers than the wireline environment. In wireline networks, packet loss is mainly due to congestion and consequent buffer overflow at routers, and the amount of congestion and the behavior of transport protocols are the main determinants of throughput. In wireless networks the dominant types of losses, radio losses and interference losses, are due to PHY and MAC layer phenomena. Radio losses occur due to high bit error rates (BER) caused by poor channel quality. Interference losses occur when multiple sources within range of each other transmit at the same time in the same frequency band. Hence, in addition to congestion and transport protocol behavior, wireless throughput depends on PHY and MAC layer behavior and PHY and MAC layer mechanisms to prevent losses. The presence of a larger number of factors that effect throughput make wireless throughput prediction a more challenging problem than wireline throughput prediction. The fact that PHY and MAC layer phenomena are the main determinants of wireless throughput means that the tools used for measuring path properties and predicting throughput in wireline networks, such as our SVR-based method discussed above, cannot be applied to wireless networks because the assumptions they are based on do not hold.

There have been a large number of efforts to model and predict wireless throughput [47, 97, 105, 103, 68, 92, 80]. The primary challenges these efforts have faced have been those of analysis comprehensiveness and practicality. Given the large number of complex factors effecting wireless throughput, the earliest of the above-listed efforts [47] considered only interference-free broadcast traffic, and analysis comprehensiveness increased with each model, with the latest and most complete, [80], handling an arbitary number of sources of interference interference, unicast traffic, and TCP. Hence achieving analysis comprehensiveness for wireless throughput modeling and prediction has been a long and laborious process. Practicality has been a challenge on two accounts. The first is system opaqueness, and the second measurement delay. System opaqueness is a challenge because these approaches assume knowledge of the network topology such as the number of interfering sources, and assume cooperation of the other nodes in the network to take pairwaise measurements of channel quality and loss rates between nodes. These assumptions are not realistic in the vehicular opportunistic networking scenario because the vehicular clients are autonomous. The measurement delay of the above approaches is prohibitive under the stringent time constraints of the vehicular environment because the number of measurements is quadratic in the number of nodes in the network.

We develop an SVR-based technique for predicting wireless TCP throughput using a set of short active measurements to gauge network conditions. Our measurements are conducted between the target client and the AP, and do not involve any other nodes in the network, so they surmount the system opaqueness challenge. The measurements are short, 0.3–1.25 seconds, so they overcome the measurement delay challenges faced by earlier approaches. A major strength of our approach is the ease with which it handles analysis comprehensiveness compared to prior work. Our combination of simple and short active measurements and SVR-based prediction allows us to predict TCP throughput for fully general wireless networks whereas the prior work, which combines analytical models and seed measurements to predict throughput, had to go through a long process of building incrementally more complete models to achieve analysis comprehensiveness.

The accuracy and timeliness of our approach is cause for cautious optimism. The accuracy is significantly lower than that for the wireline case. In most of our wireless experiments only 10% to 30% of predictions are within 10% of actual throughput, while in our wireline experiments 50% or more of predictions are within 10% of actual, a drop from wireline to wireless of a factor of 2 to 5. However, **80% to 100% of predictions are within a factor of two of actual throughput**. The factor of two bound on accuracy means that predictions are useful for many (but not all) applications. **This bound on accuracy can be achieved using measurements lasting for as little as 0.3 seconds, and for vehicular clients moving at speeds of 15–25 mph**, so our approach is suitable for practical deployment in opportunistic wireless networks.

The third problem to which we apply the SVM framework is the identification or *fingerprinting* of the rate adaptation algorithms deployed in an 802.11 environment. Unlike the first two regression-based throughput prediction problems, this is a classification problem.

802.11 rate adaptation algorithms have a large impact on wireless throughput. However, to date, they have been studied in isolation rather than as a component of production wireless network performance studies. Such studies are based on passively acquired network traces because clients in a wireless network are autonomous, *i.e.*, are not under the centralized control of network administrators, so their cooperation in any performance analysis effort cannot be assumed. There is currently no way to understand the impact of rate adaptation algorithm behavior on the performance of production networks because there is no practical (*i.e.* passive trace based) technique to identify the deployed rate adaptation algorithms.

We develop an SVM-based method to identify the 802.11 rate adaptation algorithms deployed in a network using only passive traces. 802.11 rate adaptation algorithms can exhibit a very large number of rate change permutations depending on the prevailing network conditions and the resulting packet loss patterns. Consequently, attempting to identify algorithms by explicitly enumerating all possible rate change patterns would be extremely tedious if not impossible. However, learning-based methods are suited to such problems, so we use SVM for rate adaptation algorithm identification.

Our SVM-based approach achieves a classification accuracy of 95%-100%. The key to our method's high accuracy is careful feature selection and classifier construction based on a broad sampling of algorithm behavior under a wide range of operating conditions (Chapter 6 Section 6.4). We also show that the classifiers generated by our method are portable.

The application of SVM to the identification of 802.11 rate adaptation algorithms demonstrates SVM's ability to accurately and efficiently model complex relationships between large numbers of variables. We have shown in Chapter 6 Section 6.4 that a set of approximately 4000 input features is necessary for 95%-100% accuracy. Such a large feature set is a fundamental requirement for correct algorithm identification and not a quirk of our methodology because, as discussed in Chapter 6 Section 6.4, algorithm behavior at different time granularities much be considered to facilitate accurate classification. Hence a learning-based method such as SVM that can handle complex relationships between large numbers of variables effectively is not merely desirable but essential for 802.11 rate adaptation algorithm classification.

Our SVM-based 802.11 rate adaptation algorithm fingerprinting method meets the analysis comprehensiveness requirement because it identifies algorithms implemented by a commodity 802.11 driver (MadWifi). The SVM framework provides for extensibility and maintenance because input features can be added easily and efficiently, so when new algorithms are developed, classifiers can be updated with minimal effort. Our careful selection of input features makes our method robust for a wide range of operating conditions. Since classifiers are portable, pre-computed classifiers can be used to eliminate any start-up or retraining latency, so our method has high agility. Our method is practical because it is based entirely on passively acquired packet traces and does not assume any cooperation from the wireless clients in a network.

1.3 Major Contributions

This dissertation makes the following main contributions:

- We present a methodology for TCP throughput prediction over wireline paths using Support Vector Regression (SVR), a state-of-the-art machine learning technique. Our methodology is 1.6 times more accurate than the best technique in prior work, predicts throughput for short as well as bulk transfers while prior work considers only bulk transfers, handles level shifts in network conditions with greater agility, and introduces significantly less measurement traffic into the network. We implement our methodology in a tool called PATHPERF, and test it on a diverse set of wide-area paths to show that the strengths of our methodology hold for wide-area paths in addition to the laboratory environment.

- We develop an SVR-based mechanism for TCP throughput prediction over wireless paths. Unlike prior methods that require active measurements involving all senders and receivers in a wireless network for throughput prediction, our method can predict throughput based entirely on measurements taken from the node that desires the throughput prediction. Our method can generate predictions using measurements as short at 0.33 seconds, rapidly enough for them to be useful for mobile drive-through wireless applications. 80–100% of predictions generated by our method are within a factor of two of actual throughput, even for nodes driving at speeds of 15–25mph.

- We introduce a technique for identifying the 802.11 rate adaptation algorithms deployed in a network using Support Vector Machines (SVM), which has an accuracy of 95–100%. Our technique is based entirely on the use of passive network traces. The application of SVM/SVR to rate adaptation algorithm identification is different from its application to the two throughput prediction problems because in the latter case the feature set is small and well-known or obvious while in the former case it is non-obvious and turns out to be almost 4000 to support high accuracy. To our knowledge, ours is the only effort in the literature to date that attempts to identify rate adaptation algorithms.

1.4 Thesis Organization

In Chapter 2, we present prior work related to each of the three applications of SVM, and place our contributions in the context of the prior work.

In Chapter 3, we provide an overview of the aspects of Support Vector Machines and Support Vector Regression relevant to this dissertation.

In Chapter 4, we present an SVR-based solution to the problem of TCP throughput prediction for wireline networks. Our solution is significantly more accurate, robust, and agile compared to the best solution in prior work, and also introduces considerably less measurement traffic into the network.

In Chapter 5, we present an SVR-based solution to the problem of TCP throughput prediction for wireless networks. Unlike earlier solutions, our solution does not assume any cooperation from senders and receivers other than the node desiring the prediction. Our predictions can be generated with reasonable accuracy in under 0.5 seconds, allowing them to be useful enough for mobile drive-through wireless applications.

In Chapter 6, we present an SVM-based solution to the problem of identifying the rate adatation algorithms deployed in a wireless network based entirely on passive network traces. Our identification methodology has an accuracy of 95%–100%.

In Chapter 7, we summarize the contributions of this dissertation and identify directions of future work.

Chapter 2

Related Work

In this chapter, we present studies related to each of the three problems addressed in this dissertation. A broader overview of the use of Support Vector Machines for solving problems in computer networking is presented in Chapter 3 Section 3.7.

2.1 Related Work for Wireline TCP Throughput Prediction

There are four areas of research related to our work on SVR-based wireline TCP throughput prediction. The first area is Internet measurement studies that demonstrate that TCP throughput distributions over the Internet have certain characteristics in common – if TCP throughput was completely random, all efforts at predicting it would be doomed to failure. The second and third areas are two different approaches to TCP throughput modeling and prediction, the formula-based approach and the history-based approach. The formula-based approach *(FB)* represents TCP throughput as a mathematical function based on network path properties such as loss rate and RTT, and TCP features such as congestion window size and acknowledgment schemes. The history-based approach *(HB)* predicts future throughput based on time-series analysis of past throughput. The fourth area of related work is tools for active measurements of network path properties such as loss rate and available bandwidth. The existence and accuracy of these tools is an essential component of our approach because it allows our approach to be transferred from a controlled laboratory environment equipped with ground-truth passive monitoring facilities to Internet paths without any such facilities. In what follows, we first present in detail the related work from each of these four areas, and then discuss the new contributions of our work.

2.1.1 Internet Performance Measurement Studies

A number of measurement studies of the Internet have been carried out to investigate how performance metrics such as delay, loss, and throughput vary over time for given Internet paths. Such studies are of great importance for any attempt to predict network performance because they provide insight into whether any predictability exists in network performance in the first place.

Balakrishnan *et al.* in [16] found that the throughput of individual hosts over time can be represented as a lognormal distribution. They found *temporal locality* of throughput, *i.e.*, for about 90% of hosts in the study, throughput for successive transfers differed by a factor of two or less. The also found *spatial locality* of throughput, *i.e.*, that throughput for nodes close to each other in terms of network distance was quite similar. Barford and Crovella's application of critical path analysis to TCP file transfers provides a more detailed perspective on how delay, queuing and loss relate to TCP performance [17].

Paxson in [100] studied the behavior of loss, delay, and bottleneck bandwidth on Internet paths and found the following: *(a)* loss was bursty, with bursts typically being longer than 200ms, as a result of drop tail queues at routers, *(b)* delay was usually between 0.1 and 1 second and paths were very asymmetric in terms of delay, and *(c)* bottleneck bandwidth between a pair of hosts was generally stable over time.

Zhang *et al.* in [142] introduce *constancy* to represent more precisely the degree of stability of loss, delay, and throughput in Internet paths. They consider *mathematical constancy*, which exists if a set of measurements can be described by a single, time-invariant mathematical model, *operational constancy*, which determines the granularity of change of a metric needed to have a meaningful impact on application performance, and *predictive constancy*, which indicates whether time-series analysis of past measurements is useful for predicting future values. For throughput, they find that constancy of all three forms exists on the order of minutes but not hours.

Finally, Zhang *et al.* in [141] study the distribution of rates at which flows transmit data. They find that the flow rates have a skewed distribution, but it is not as heavily skewed as the distribution of flow sizes. They also find that flow rates and flow sizes are correlated, suggesting that the amount of data transferred is not intrinsically determined, but rather dependent on the performance a user can get from the network.

2.1.2 Analytical Models of TCP Throughput

Since the seminal work by Jacobson and Karels established the basic mechanisms for modern TCP implementations [59], it has been well known that many factors affect TCP file transfer performance. In general, these include the TCP implementation, the underlying network structure, and the dynamics of the traffic sharing the links on the path between two hosts. A large number of research efforts have modeled TCP throughput using analytical formulas which take into account some of the above-mentioned factors, with later models attempting to improve the accuracy of the earlier ones. The presence of a large number of efforts to derive analytical formulas for modeling TCP throughput is a testament to the difficulty of capturing the dynamics of TCP.

In [96], Ott *et al.* derive the stationary distribution of the TCP congestion window size by approximating the discrete-time process of window size evolution with a continuous-time process. They consider only the congestion avoidance phase, assume that losses are independent and that loss is always signaled by triple duplicate ACKs. They were the first to derive the famous inverse square root relationship between throughput and loss, *i.e.*, to show that throughput is proportional to $\frac{1}{\sqrt{loss\ rate}}$. Mathis *et al.* in [84] present an approximate version of the model in [96] based on the assumption that loss is periodic, and show that throughput is proportional to $\frac{maximum\ segment\ size}{RTT*\sqrt{loss\ rate}}$. Misra *et al.* in [87] improve on the model in [96] by representing loss as a function of instantaneous window size instead of treating it as a constant.

In [75], Lakshman *et al.* model high delay-bandwidth product networks, *i.e.,* networks where the delay-bandwidth product is the same or higher order of magnitude as the amount of buffering on the bottleneck. A high delay-bandwidth product is generally a property of wide area paths rather than LANs. They model TCP slow start as a discrete time process and congestion avoidance as a continuous time process similar to [96], and show that for high delay-bandwidth product networks, throughput is proportional to $\frac{1}{loss\ rate\ *\ (delay\ *\ bandwidth)^2}$. And while earlier studies had identified the bias of TCP against high delay-bandwidth product paths, Lakshman *et al.* were the first to present a quantitative analysis of the bias. Kumar *et al.* in [72] build a model similar to that of [75] but focus instead on LANs and add coarse-grained timeouts to the analysis.

Padhye *et al.* in their model of TCP Reno performance [98] consider losses signaled by both triple duplicate ACKs and timeouts. They introduce the concept of *rounds* for modeling TCP loss. A round starts when the current window size worth of packets is sent and ends when the first ACK is received. Losses within a round are assumed to be correlated to capture burstiness due to drop tail queues, and losses in different rounds are assumed to be independent. Cardwell *et al.* in [31] extend the Padhye model by adding connection establishment and slow start latencies.

Abouzeid *et al.* in [10] present a comprehensive stochastic model of TCP loss, taking into consideration both random and correlated loss and small and large delay-bandwidth products. Altman *et al.* in [12] model TCP performance assuming that loss events have a general distribution, *i.e.,* the only assumption regarding the loss process is that it is stationary. They show that for a general distribution, throughput is proportional to $\frac{1}{RTT\ *\ \sqrt{loss\ rate}}$ just like it is for specific loss event distributions considered in earlier works discussed above.

All models presented so far assume that TCP packet traces are available to compute end-to-end per-flow metrics such as loss rate and RTT. Goyal *et al.* in [48] consider the case where the only available information is via SNMP MIBs from routers at the granularity of the aggregate traffic through the routers rather than individual flows. They adapt the Padhye model [98] to handle metrics derived from SNMP MIBs. Arlitt *et al.* in [14] present a heuristic based on the Cardwell model [31] for predicting the throughput of a TCP transfer based on the transfer size and the RTT of the first packet exchange of the transfer.

2.1.3 History-based TCP Throughput Prediction

A number of studies have used time-series analysis of past TCP throughput on a path to predict future throughput. These are referred to as history-based prediction approaches.

Vazhkudai *et al.* in [128] predict throughput for large files using mean-based, median-based, and auto-regressive predictors and find that the mean-based and median-based predictors did as well as the more heavyweight auto-regressive predictors. Lu *et al.* in [81] note the strong correlation between TCP throughput and transfer size. Their goal is to exploit this relationship for throughput prediction for arbitrary file sizes without having to build a history by transferring every possible file size. They accomplish this by conducting one small transfer and one large transfer, and predicting throughput for a given size by extrapolating based on the throughput of these two transfers. Similar to our work, Lu *et al.* find that prediction errors generally followed a normal distribution.

A problem with history-based approaches is that they introduce a large amount of measurement traffic into the network, because in order to predict the throughput of a transfer of a given size, at some point in the past a file of comparable size has to be transferred to establish the historical throughput. Most history-based methods perform some type of averaging, so in general the file has to be transferred a number of times. Given changes in path conditions over time, recent history is more likely to yield accurate predictions than distant history. The need for greater accuracy calls for conducting transfers frequently and as close to prediction time as possible, while the need to lower overhead calls for conducting transfers infrequently. For the Network Weather Service [120], Swany *et al.* handle the competing demands of accuracy and low overhead by using a combination of large, heavyweight transfers, conducted infrequently, and small, lightweight transfers, conducted frequently.

He *et al.* in [54] consider three different ways of predicting TCP throughput, *(a)* using available bandwidth as a proxy for, and hence predictor of, throughput, where available bandwidth is defined as the spare capacity on a network path, *(b)* using analytical formulas, specifically the Padhye formula from [98], and *(c)* time-series based predictors such as EWMA and Holt-Winters. They find that available bandwidth is a poor proxy for throughput, because *(a)* available bandwidth is an inelastic measure of link conditions while TCP throughput is an elastic quantity, varying with the degree of multiplexing on the link, *(b)* available bandwidth measurements take far too long to converge to be useful for predicting the throughput of short TCP transfers, and *(c)* there is no obvious way to account for slow-start effects with available bandwidth measurements. They show that using analytical formulas for throughput prediction yields very poor accuracy for wide area paths. One important reason for the inaccuracy of formula-based predictors is that analytical formulas are designed to *model* rather than *predict* throughput, *i.e.,* that the formulas assume that the values of variables such as loss and RTT are obtained *during* the flow of interest, using packet-level passive traces of the flow. However, to use the formulas for prediction, loss and RTT values have to be obtained prior to the start of the flow whose throughput is to be predicted. Active measurements have to be used to acquire loss and RTT values because there is no data traffic and therefore no passive traces prior to the beginning of the flow. Predictions are hence inaccurate for two reasons, *(a)* measurements of loss and RTT are made prior to, rather than during, the flow, so inaccuracy is introduced because the values are unlikely to be exactly the same before and during the flow, and *(b)* measurements are made using UDP-based active measurement tools so the loss and RTT values they observe will most likely be somewhat different from what a TCP flow, with its window-based congestion control, will experience. He *et al.* also show that history-based predictors such as EWMA and Holt-Winters yield significantly better accuracy than formula-based predictors. They find that increased multiplexing on a link increases history-based predictor accuracy, but high link utilization and throughput level shifts and outliers decrease accuracy.

2.1.4 Practical Tools for Measuring Internet Path Properties

As described in more detail in Chapter 4 Section 4.3, our TCP throughput prediction technique uses active measurements of path properties such as loss, RTT, and available bandwidth to assess the conditions on a given path.

A complete overview of techniques for measuring loss, RTT, and available bandwidth is beyond the scope of this thesis – an excellent summary can be found at [43], and [112] present a comparison of available bandwidth measurement tools. We chose Badabing [115] and Yaz [116] for loss and available bandwidth measurements respectively because they were the most accurate tools available to us. However, the Support Vector Regression framework is flexible, so if more accurate path measurement tools become available in the future they can trivially replace the existing tools to yield potentially more accurate predictions.

2.1.5 Our Contributions Relative to Prior Work

As discussed in Chapter 1 Section 1.1, high accuracy is the most important criterion for evaluating any network performance analysis method. However, a number of other criteria, such as analysis comprehensiveness, extensibility and maintenance, robustness, agility and practicality are important as well. For wireline TCP throughput prediction, practicality translates to low measurement traffic overhead and use of practical techniques for measuring path properties, agility to quick response to level shifts, analysis comprehensiveness to the ability to extrapolate and predict throughput accurately for previously unseen file sizes, robustness to high prediction accuracy over a diverse set of real wide area paths, extensibility and maintenance to the ability to handle different flavors of TCP without having to rebuild a predictor from scratch.

Our SVR-based predictor is at least as good as, and often outperforms, the better of the history-based and formula-based predictors on all of the above criteria. In the discussion below, we compare the SVR-based predictor with both formula-based and history-based predictors on all of the above criteria. However, it is the comparison with history-based predictors that is of real importance because [54] has already shown history-based predictors to be significantly more accurate than formula-based ones. We discuss formula-based predictors merely for the sake of completeness.

The most important feature of a TCP throughput prediction mechanism is high accuracy, because accurate predictions allow applications to make good decisions (*e.g.,* accurately selecting the highest throughput path in an overlay network) and improve performance. We evaluate the accuracy of our prediction mechanism in two ways, using *oracular* and *practical* measurements of path properties as described in Chapter 4 Section 4.3. For bulk transfers in heavy traffic (90% average utilization on the bottleneck link) using *oracular* measurements, TCP throughput is predicted within 10% of the actual value 87% of the time, representing nearly a 3-fold improvement in accuracy over history-based methods. For practical measurements of path properties, predictions can be made within 10% of the actual value 49% of the time, an improvement of a factor of 1.6 over history-based methods.

The SVR-based predictor introduces a significantly lower amount of measurement traffic compared to history-based methods. As described in Chapter 4 Section 4.5, a typical run of the SVR-based predictor will generate a factor of 13 less measurement traffic than a history-based predictor. There are two reasons for this. First, the SVR-based predictor uses lightweight active measurement tools instead of bulk TCP transfers to gauge path conditions. The second reason is the set of path properties that the SVR-based predictor needs for high accuracy. We consider three path properties as inputs to the SVR, RTT, loss rate, and available bandwidth. We find that high prediction accuracy can be achieved by using only RTT and loss rate, and the addition of available bandwidth to the input features does not yield any increase in prediction accuracy (Chapter 4 Section 4.3). This is good news, because while active measurements of RTT and loss rate are lightweight and introduce little measurement traffic, active measurements of available bandwidth are heavyweight. The fact that these heavyweight active measurements can be excluded without compromising accuracy helps reduce the measurement overhead of SVR-based prediction. The measurement overhead of SVR is the same as that of formula-based predictors because the latter also require RTT and loss rate measurements.

[54] has shown that *level shifts* in path conditions pose a challenge for history-based predictors. The use of long-term history in history-based predictors has the advantageous effect of smoothing out predictions by preventing ocassional outliers from introducing errors into predictions, but at the same time has the disadvantageous effect of making them slow to respond to level shifts. The SVR-based predictor does not suffer from this problem. As shown in Chapter 4 Section 4.3, it is extremely responsive to level shifts and can generate accurate predictions after observing one single training sample at the new throughput level. Formula-based predictors are the most responsive to level shifts in theory. Both history-based and SVR-based predictors need one or more training samples at the new throughput level, but formula-based predictors require none (they require only the new values of RTT and loss) because the formula represents the complete spectrum of possible throughput behavior and there is no learning component to predictions. However, the inaccuracy of formula-based predictors in practice renders their responsiveness immaterial.

History-based predictors are focused on bulk transfers, *i.e.,* transfers large enough that start-up effects such as connection setup and TCP slow-start have a negligible impact on throughput. There is no direct way to extrapolate from one file-size to another for history-based methods ([81] indirectly extrapolate for file sizes of 400KB or larger), and the only way to generate accurate predictions for TCP transfers where start-up effects do impact throughput is to explictly construct a new predictor for the different transfer sizes. SVR-based predictors, on the other hand, can directly extrapolate over a wide range of transfer sizes with high accuracy from a small number of transfer sizes in training. As illustrated in Chapter 4 Section 4.3, we found that a training set of only three file sizes results in accurate throughput predictions for a wide range of file sizes. Formula-based predictors are, once again,fundamentally different from both history-based and SVR-based predictors because they have no learning component and the formula represents the complete spectrum of possible throughput behavior at all transfer sizes.

The SVR-based predictor is accurate not only in laboratory settings but also over a diverse set of wide area paths. We tested the SVR-based predictor in over 18 paths in the RON testbed [13]. 2 of the 18 paths were trans-European, 9 were trans-Atlantic, and 7 were trans-continental-US. The RTTs varied from 8ms to 145ms. Chapter 4 Section 4.4 shows that the SVR-based predictor maintains high accuracy for most of these paths, and explains the reasons for decreased accuracy over certain wide-area paths. The accuracy profile of the history-based predictor follows that of the SVR-based predictor. We do not consider formula-based predictors in our wide area experiments because [54] has already shown formula-based predictor accuracy to be quite poor for wide area paths.

Both SVR-based and history-based predictors are more flexible than formula-based predictors in terms of handling different flavors of TCP. The fundamental difference between formula-based predictors on the one hand and history-based and SVR-based predictors on the other is that history-based and SVR-based predictors have a learning component while for the formula-based predictors the complete range of throughput behavior is captured by the formula itself. The absence of a training phase, and the absence of associated delay and measurement traffic cost is clearly an advantage of formula-based methods. However, the price of this advantage is a lack of flexibility for formula-based methods. Formula-based predictors are specific to flavors of TCP, and a separate formula has to be explicitly derived for each flavor of TCP, making formula-based predictors expensive to maintain.

Over the last few years, online tools such as *Speedtest* [7] and M-Lab's *Network Diagnostic Tool* [5] have become available. These allow end users to measure TCP throughput achievable via their ISPs, and to identify throughput bottlenecks. Such tools differ from our work in two ways. First, they measure current throughput while our work predicts future throughput. Second, their throughput measurements are used for relatively long time scale tasks such as comparing ISP performance or identifying bottlenecks to improve network provisioning and configuration. In contrast, our throughput predictions are used for shorter time scale tasks such as allowing applications to take advantage of higher throughput paths in overlay networks or determining a new TCP-friendly transmission rate as path conditions change.

2.2 Related Work for Wireless Throughput Prediction

Areas of work related to our research on TCP throughput prediction for opportunistic wireless networks include *(a)* studies on wireless network performance, *(b)* studies on opportunistic networks, and *(c)* studies on 802.11 access point selection. In this section, we present prior work from each of these three areas and discuss our contributions relative to the prior work.

2.2.1 Studies on Wireless Network Performance

There is an extensive body of work on wireless network performance. Modeling and predicting wireless network performance is extremely challenging because link quality, signal propagation and interference are very deployment and location specific and hence nearly impossible to model analytically. Early efforts to model wireless network performance assume abstract models of signal propagation and interference. Later efforts seed throughput models with deployment-specific measurements of interference. We discuss each body of work in turn.

2.2.1.1 Abstract Models of Wireless Network Performance

There are two sets of abstract models of wireless network performance, those that relate to general wireless networks and those that are specific to 802.11-based wireless networks.

In their seminal paper [51], Gupta and Kumar study the capacity of general ad hoc wireless networks with no centralized control, no hidden terminals and collision losses, and multi-hop routing from source to destination via intermediate nodes. They model successful packet transmission rates based on interference at the sender (carrier sense), ambient noise and the distance between the sender and receiver. They show that for a network with n randomly-placed nodes, the average per-node throughput is proportional to $\frac{total\ channel\ capacity}{\sqrt{nlogn}}$ and for a network with n optimally-placed nodes, the per-node throughput is proportional to $\frac{total\ channel\ capacity}{\sqrt{n}}$.

The main result of [51], that per-node throughput decreases as the number of nodes in the network increases, is discouraging for large ad hoc networks, and argues for smaller ad hoc networks or infrastructure networks to reduce packet routing overhead. Later projects identify ad hoc deployment scenarios where this bound on throughput can be improved. Grossglauser *et al.* in [49] show that if nodes are mobile and data transmission can tolerate delays of minutes to hours, throughput can be made independent of the number of nodes in the network by transmitting data only when the source and destination are close to each other, thus cutting the multi-hop routing overhead. Li *et al.* in [79] show that for certain realistic non-random traffic patterns network throughput can exceed the bound derived for random traffic patterns in [51]. Gastpar *et al.* in [45] show that if there is a single source and destination in an ad hoc network, and the remainder of the nodes merely act as relays, throughput can be made independent of the number of nodes with use of appropriate network coding. Jain *et al.* in [60, 61] use the interference model from [51] to determine whether a given traffic matrix is feasible for a given network by modeling the network as a conflict graph. They show that maximizing throughput in a network is NP-hard, so they present heuristics for the task. Kumar *et al.* in [73] improve on the work in [60, 61] by designing heuristics that allow network throughput to be within a constant factor of the maximum possible throughput.

There are also a large number of analytical models focused specifically on predicting the performance of 802.11-based wireless networks. Bianchi [23] analytically predicts system throughput for 802.11 networks that use either the distributed coordination function (DCF) or RTS/CTS for media access control based on 802.11 MAC parameters such as minimum and maximum backoff windows. The throughput predicted is the *saturation* throughput under ideal channel conditions, *i.e.*, no hidden terminals and capture effects. The *saturation* throughput is defined as the stable throughput reached by a network of a finite number of senders as offered load increases to the point where every node always has a non-empty transmission queue. Throughput is calculated based on packet collision probability, which in turn is calculated based on the simplifying assumption that at each transmission attempt, regardless of the number of previous failed transmissions, each packet experiences collisions with a constant and independent probability. Burmeister *et al.* in [28, 27] model 802.11 throughput for the AP-client communication model instead of the ad hoc network model. They are the first to consider the impact of 802.11 rate adaptation on throughput. They assume that wireless signal strength decreases with distance. They show that in the presence of even a single distant receiver, total network throughput can decrease dramatically because successful packet delivery to the distant receiver occurs at lower transmission rates, decreasing the medium access time available to all other nodes.

2.2.1.2 Hybrid Models of Wireless Network Performance

Studies such as [97, 105] show that abstract RF propagation models, based on assumptions such as the interference range of a sender being twice the successful transmission range, are inaccurate in practice. The failure of abstract models of interference has led to *hybrid* models, which consist of an analytical throughput model that is seeded with deployment-specific interference measurements in place of abstract interference models. The goal of measurement-based models is to categorize the throughput variation of a target network link with sender *A* and receiver *B* for all pairs of links *AB* for all possible combinations of interference from other senders *S* in the network. The number of possible interference combinations increases exponentially with the number of senders in the network, making a complete set of measurements to seed the model impractical. All the hybrid models described in this section approximate the interference characteristics of the network with a smaller number of measurements.

Gopalakrishnan *et al.* in [47] use a combination of environment-specific RSSI measurements to seed signal propagation models and simulations to predict 802.11 throughput in an interference-free environment, *i.e.*, an environment where packet loss is due to bit errors only. Padhye *et al.* in [97] use pairwise broadcast link interference to approximate the full spectrum of possible unicast network interference combinations using $O(n^2)$ measurements, where n is the number of senders, and report that model accuracy decreases when *(a)* 802.11 rate adaptation is used for unicast traffic because there are no acknowledgments and hence no rate adaptation for broadcast, and *(b)* when there is a time lag between model seed measurements and model testing. Reis *et al.* in [105] model throughput for the case of broadcast transmissions and binary interference (only two senders transmitting simultaneously in the network) using $O(n^2)$ measurements that can be acquired in $O(n)$ time. Kashyap *et al.* in [68] model throughput for both broadcast and unicast traffic for an arbitrary number of interfering sources using $O(n^2)$ seed measurements by representing 802.11 protocol states as a discrete-time Markov model. Qiu *et al.* in [103] also model the 802.11 DCF using a Markov chain and $O(n^2)$ seed measurements. Their model predicts throughput for both saturated and unsaturated traffic demands, and they optimize the model by pruning model states that are very unlikely to occur. Niculescu [92] approximates interference by assuming independence of interference effects, *i.e.,* by assuming that the throughput of a link *AB* with n simultaneous sources of interference S_1–S_n is the product of the throughput of *AB* when each interferer S_i interferes with *AB* individually. Li *et al.* in [80] further reduce the complexity of earlier models by constructing a model that assumes the independence of collision and error loss in addition to the independence of interference, and handles RTS/CTS, multi-hop networks, and TCP in addition to CBR traffic.

2.2.2 Opportunistic and Vehicular Wireless Networks

In this section, we present an overview of opportunistic wireless networks because they are the motivation for our work on wireless TCP throughput prediction. Our work targets opportunistic wireless networks, particularly opportunistic vehicular networks, because they present the most challenging scenario for wireless throughput prediction due to their opaqueness and stringent prediction speed requirements, as discussed in Section 2.2.4.

Opportunistic wireless networking is a communication model in which clients passing through the range of one or more open APs gain temporary network connectivity. Opportunistic networking has been made possible by the growth in popularity of WiFi access over the past decade, which has resulted in dense deployments of 802.11 APs in urban areas. The clients in an opportunistic network can be either stationary or mobile and vehicular.

A common example of a stationary opportunistic network is *Fon* (www.fon.com). *Fon* supports a *cooperative* opportunistic networking model. *Fon* members open up their home APs to allow access to other *Fon* members who happen to be in the vicinity and desire connectivity in exchange for similar guest access via other members' home APs. The goal is to provide *Fon* members WiFi connectivity worldwide via such bandwidth exchange.

There has been extensive research into 802.11-based vehicular opportunistic networks over the past few years. The earliest efforts investigated the feasibility of communication between fixed APs and moving vehicles, because 802.11 had not been designed for this purpose [1]. Ott and Kutscher [94] study the communication between vehicular clients and stationary roadside APs on a German Autobahn in an environment free of 802.11 interference sources. The vehicular clients move at 80-180 km/h, and are equipped with external antennas. They find that the connection between the vehicular client and the AP consists of the *entry*, *production* and *exit* phases. Connectivity is poor, *i.e.*, losses are high, in the *entry* and *exit* phases. Connectivity is best during the *production* phase, which is typically a 200m range around the AP where most of the data is transferred. For speeds of 80-180 km/h, the *production* phase lasts for 4–9 seconds and allows for the transfer of 1.5–5.0MB of data for both TCP and UDP. This volume of data transfer is likely to be sufficient for a number of applications. Gass *et al.* [44] study communication between vehicular clients without external antennas and roadside APs in an environment free of 802.11 interference sources. They consider transport and application layer effects in addition to link layer effects. They find, just as [94] did, that for driving speeds ranging from 5–75 mph, bulk transfers for both TCP and UDP perform quite well. However, performance is considerably poorer for HTTP because small-sized HTTP request packets incur full round-trip delays without transferring useful application data in the short vehicle-AP connectivity window. Bychkovsky *et al.* in the *CarTel* project [29] study the behavior of vehicular 802.11 clients driving through the Boston and Seattle metropolitan areas at speeds of up to 60 km/h attempting to connect to open APs. They find that the median duration of link layer connectivity is 13 second, the mean time between successful associations is 75 seconds, the median throughput is 30KBytes/s, and the average connection can transfer 216KBytes of data in this setting. They also find that associations are equally likely

[1] Cellular networks support communication between vehicles and fixed base stations, but 802.11 offers significantly higher bandwidth and has the advantage of being part of the unlicenced spectrum

across a range of speeds. Mahajan *et al.* in [83], in an urban environment similar to that of [29], find that the well-defined entry, production, and exit phases observed in [94] on autobahns do not hold in urban environments, and there are abrupt losses of connectivity even in the production phase. However, they found that connectivity behavior at particular locations tends to be predictable. Hadaller *et al.* in [52] find several causes of sub-optimal performance in communication between vehicles and APs, such as losses in the entry phase leading to backoffs that last into the production phase, lengthy AP scanning and selection, poor MAC layer rate selection. The *DieselNet* project studies WiFi communication between buses and between buses and fixed infrastructure in Amherst, MA., and has developed ad hoc routing protocols for efficient communication in the face of intermittent connectivity [26, 140]. Deshpande *et al.* in [39] have compared the performance of 3G cellular connectivity and WiFi connectivity in the vehicular scenario, and provide evidence that a hybrid network is the best option for a combination of high connectivity, high throughput and low cost. The consensus of all these studies is that while there are challenges, 802.11-based communication between APs and moving vehicles is feasible, *i.e.*, is adequate for supporting applications that can tolerate some amount of delay.

The challenge of vehicular opportunistic networks is that connectivity is *fleeting* and *intermittent*. The good news is that while connectivity is established communication occurs at high speed. Mitigating the effects of fleeting or short-lived connections requires making use of connectivity time as efficiently as possible. *QuickWiFi* [40] is an implementation that reduces client to AP connection time from about 10 seconds to 400ms by optimizing channel scanning and eliminating MAC-layer delays and timeouts – 10 seconds is acceptable connection setup latency for a stationary client, but for a vehicular client it leaves no time for useful data transfer. Mitigating the effect of intermittent connectivity requires saving and restoring transport layer connection state as connectivity comes and goes. Ott and Kutscher in [95] and Eriksson *et al.* in [40] implement persistent *sessions* using client and server side proxies. Unlike TCP connections, sessions persist when connectivity is lost, and restore transport layer connections when connectivity returns. The difference between the two solutions is that [95] implement a session layer that runs on top of TCP, while [40] implement a combined transport and session layer that, in addition to providing connection loss recovery, conducts wireless-aware congestion control.

A different approach to making the most of fleeting and intermittent connectivity in vehicular opportunistic networks is to enable applications to reduce the amount of data being sent, or to prioritize the transmission of the most important data. A natural use of vehicular networks is for collecting data about driving conditions, *e.g.*, road traffic congestion reports to facilitate route planning [124, 123] or reports of damage to road surfaces to facilitate repair work [6]. This sensor data needs to be transferred to fixed infrastructure either directly or via other *data mule* vehicles when there is a connectivity window. To make efficient use of the connectivity window, Hull *et al.* in [58] present *CafNet*, a callback-based network stack. Unlike traditional network stacks that buffer and then stream application data in the FIFO order, *CafNet* does not buffer any data. Instead, it informs the application when connectivity is available, allowing the application to prioritize which data should be sent at connectivity time rather than making the application commit data to the network in FIFO order in advance.

Even though all the studies discussed so far in this section focus on using 802.11a/b/g for vehicular wireless communication, and there is strong evidence that these protocols can support useful applications in the vehicular environment, they were not designed to support vehicular communication. The 802.11p Dedicated Short Range Communication standard is being developed specifically for communication between neighboring vehicles to allow exchange of information such as vehicle GPS coordinates, velocity and road conditions to determine if accidents are likely. The 802.11p physical layer is designed to handle the challenges of high speed mobility, such as severe signal fading. Bai *et al.* in [15] investigate the behavior of 802.11p PHY in a practical setting.

2.2.3 Access Point Selection for 802.11 Networks

When a wireless client is in an environment with multiple APs, it has to select which AP to associate with. Commodity 802.11 interface cards choose to associate with the AP that delivers the strongest signal strength. However, in many scenarios, an association decision based solely on signal strength is sub-optimal, *i.e.*, it does not result in the highest possible throughput for the client. For example, if a large number of other clients are associated with the AP with the strongest signal and contending for the AP's time , the client might achieve higher throughput by associating with a different AP with a weaker signal but fewer other clients.

A mechanism for accurate TCP throughput prediction for wireless networks can facilitate AP selection – the obvious choice would be the AP with the highest predicted throughput. There are a number of research efforts [127, 118, 91, 122, 77] that focus on AP selection based on highest projected throughput. What distinguishes these efforts from our work is that these efforts focus ranking APs by projected throughput while we focus on general-purpose wireless TCP throughput prediction that can be used by applications other than AP-selection, such as determining the TCP-friendly rate of non-TCP flows. These approaches make certain assumptions, such as the presence of beacons, so they can only be used for infrastructure networks and not for ad hoc networks, while our approach is not limited in this way. We outline these efforts in the remainder of this section.

Vasudevan *et al.* in [127] develop an algorithm for AP selection based on observing the expected and actual timing of 802.11 beacon frames. Sundaresan *et al.* in [118] propose a combination of physical and MAC layer metrics for calculating projected throughput and facilitating AP selection, but acquiring their proposed metrics requires changes to wireless interface firmware. Nicholson *et al.* in [91] focus on AP-selection based on the performance of the backhaul wireline connection to the Internet in addition to the performance of the AP-client link. Taenaka *et al.* in [122] use the number of frame retransmissions on the AP-client link to gauge link quality and guide AP selection. Lee *et al.* in [77] propose an AP selection scheme based on the behavior of packet exchanges that occur when a client scans candidate APs on different channels prior to association. And Bejerano *et al.* in [18] take a global, network-wide approach to

AP selection by pairing APs and clients based on load-balancing from the perspective of the APs and fair bandwidth allocation from the perspective of the clients.

2.2.4 Our Contributions Relative to Prior Work

In this section, we explain why there was a need for an effort such as ours for TCP throughput prediction for wireless networks in general and opportunistic wireless networks in particular. The key contributions of our approach are that (a) it does not require knowledge or cooperation of other clients in the network, and (b) it can generate predictions with useful accuracy in as little as 0.3 seconds. The remainder of this section discusses why neither the wireless performance models discussed in Section 2.2.1 nor the wireline TCP throughput prediction techniques discussed in Chapter 4 and Section 2.1 are suited to TCP throughput prediction for opportunistic wireless networks.

The wireless performance models discussed in Section 2.2.1 compromise on both the analysis comprehensiveness and practicality requirements.

The models compromise on the analysis comprehensiveness requirement by making several simplifying assumptions that do not hold for real operational networks, because modeling fully general wireless networks is extremely challenging. Common examples of simplifying assumptions include considering only binary interference, considering a single 802.11 data transmission rate instead of modeling 802.11 rate adaptation, and modeling constant bit rate traffic, *i.e.*, assuming the absence of TCP or some other transport layer protocol that supports reliability and congestion control. Our approach works for a fully general wireless network environment and makes no simplifying assumptions.

The wireless performance models from Section 2.2.1 compromise on practicality in terms of system opaqueness and scalability. The models do not handle real network opaqueness because (a) they assume complete knowledge of the number and location of interfering sources in the network and their traffic demands, and (b) they assume complete cooperation of all the nodes in the network to obtain model parameters such as pairwise link loss rate. Such cooperation and knowledge is not available in practice. The models require at least $O(n^2)$ measurements in an n-node network to obtain relevant parameter values. In an opportunistic vehicular network, the total connectivity time between a vehicle and a roadside AP is generally 10 seconds or less [40], so the throughput prediction needs to be generated in around 1 second to be useful to applications. Hence even if complete network knowledge and cooperation was available to overcome the system opaqueness problem, the measurement scalability problem would prevent the models from Section 2.2.1 from being applicable to vehicular opportunistic networks.

He *et al.* [54] and our approach in Chapter 4 present two different approaches to wireline TCP throughput prediction. Both these approaches are practical because they have been tested on real wide area paths. However, in their original forms, both approaches take tens of seconds to generate a prediction. The reason that the approach in [54] takes tens of seconds is that it considers only bulk transfers of several megabytes, which inevitably take several seconds to complete. The approach in Chapter 4 handles arbitrarily small transfers. The reason it takes tens of seconds is due to the convergence time of the active measurement tools [115, 116] use to measure path properties. Another issue is that these tools are based on assumptions that hold only for wireline environments, such as loss being an indicator of congestion whereas in wireless environments it is an indicator of interference, so they are unsuitable for use on wireless paths. Even active measurement tools designed specifically for wireless environments, such as [76, 69], take tens of seconds to converge, so they cannot be used in vehicular wireless environments

Our approach is able to generate predictions with useful accuracy in under one second in contrast to prior approaches which require tens of seconds because, as discussed in Chapter 5 Section 5.2.2 we develop a probe process based on short active measurements to gauge network path properties. Our model requires measurements between only the target client and the AP, *i.e.*, it does not explicitly take any measurements involving any other nodes in the network. Since our model requires a few short measurements that involve only the target node and the AP, and handles fully general network deployments, it handles the challenges of system comprehensiveness, scalability and system opaqueness better than both the wireless performance models discussed in Section 2.2.1 and the wireline TCP throughput prediction techniques discussed in Chapter 4 and Section 2.1.

Gerber *et al.* in [46] take a history-based approach to measuring maximum achievable TCP throughput in a 3G wireless network. Instead of using active probes, they observe application flows to estimate this metric. This work differs from our work in that maximum achievable TCP throughput is a network-centric metric that is calculated using measurements over long time scales and is used to inform network management decisions such as provisioning and configuration, while our TCP throughput predictor is a client-centric tool based on short active measurements and used to inform client decisions such as AP selection and TCP-friendly rate selection.

2.3 Related Work for Wireless Rate Adaptation Fingerprinting

There are two major types of studies related to our work on the identification of 802.11 rate adaptation algorithms. The first are studies on the design of rate adaptation algorithms. The second are studies on analyzing passively acquired packet traces that contain no explicit information to deduce the deployed applications and protocols. We present each body of work in turn, and conclude with a discussion of our contributions relative to prior work.

2.3.1 802.11 Rate Adaptation Algorithms

Given their potentially large impact on throughput in wireless networks, it is not surprising that many research efforts over the past decade have focused on improving rate adaptation algorithms. In this section, we describe the various 802.11 rate adaptation algorithms present in the literature. Our objective here is to provide a flavor for the range and complexity of algorithm behavior. The need for a learning-based classifier to distinguish between the algorithms arises because the large number of packet rate patterns that can occur with a given algorithm under different RF environments would be very difficult to enumerate explicitly.

The biggest challenge in rate adaptation algorithm design is the difficulty of determining whether losses are due to bit errors resulting from poor channel quality or due to collisions with other transmissions. In the former case, switching to a lower rate is helpful because lower rates are more robust to bit errors, and lowering the rate will decrease packet loss. However, in the latter case, switching to a lower rate is the incorrect course of action because (a) it will not reduce losses, and (b) it will make the throughput worse than it would have been at the higher rate because of the unnecessary transmission rate reduction.

Rate adaptation algorithms fall into three categories, those that use physical layer information such as signal-to-noise ratio (SNR) to trigger rate changes, those that use frame level information such as packet loss and throughput, and hybrid algorithms that combine features of both SNR-based and frame-based algorithms. SNR-based algorithms are highly responsive because SNR information is updated in a meaningful manner with every packet transmission, allowing SNR-based algorithms to adapt the rate at single packet granularity when the RF environment changes. Frame-based algorithms are slower to respond because they need to aggregate loss and throughput information over multiple packets to correctly detect a change in the RF environment. However, SNR-based algorithms have a high overhead because they assume the use of the 802.11 RTS/CTS packet exchange to gauge channel quality. SNR-based algorithms are often rendered impractical because they require changes to standard 802.11 headers to communicate channel quality information from the receiver to the sender. Hybrid algorithms attempt to combine the practicality and low overhead of frame-based algorithms with the agility of SNR-based algorithms.

Projects such as [24, 132, 30, 136] have developed frameworks for characterizing the performance of rate adaptation algorithms. They evaluate rate adaptation algorithms based on *(a)* how quickly they adapt to changes in the RF environment, *(b)* whether the rate they select is optimal for the prevailing conditions, and *(c)* whether conventional wisdom principles on which algorithm design is based do indeed hold in practice. Such efforts have yielded important insight into the relative merits of rate adaptation algorithms under a range of network conditions. Gudipati *et al.* in [50] present a packet coding mechanism that automatically handles rate adaptation. Our work is complementary to these efforts.

2.3.1.1 Frame-based Rate Adaptation Algorithms

Kamerman *et al.* in [66] present the well-known frame-based *Auto Rate Fallback (ARF)* algorithm for rate adaptation. After the first packet loss, *ARF* retransmits at the current rate, but after the second consecutive loss, it transmits at the next lower rate and starts a timer. It increases the rate to the next higher level when either the timer expires or when 10 consecutive packets are transmitted without loss. However, if the current rate is optimal (*i.e.,* the highest sustainable without loss in prevailing conditions), trying a higher rate after every 10 successful transmissions is a bad strategy because it leads to unnecessary loss. Lacage *et al.* in [74] propose a solution to this problem in the form of the *Adaptive Auto Rate Fallback (AARF)* algorithm. If, when a higher rate is tried after 10 successful transmissions and a loss results, the interval after which the higher rate is tried the next time is doubled to 20 packets and so on, up to a maximum of 50 packets. This approach attempts to balance the elimination of unnecessary losses while retaining the responsiveness to quickly take advantage of favorable RF conditions to boost throughput. Chevillat *et al.* in [33] present a similar adaptive approach to probing for higher rates. They propose having two different probing frequencies, one low and one high. Their algorithm probes at the high frequency as long as probe loss stays below a certain threshold, and switches to the lower frequency when loss exceeds the threshold.

Bicket in [24] significantly advances the state of the art in frame-based algorithms by observing that the assumption that a higher bit rate will always result in higher loss in a given environment is not necessarily true. The actual relationship between loss rate and bit rate depends on the modulation, the encoding of the rate, and the amount of noise in the RF environment. Bicket designs the *SampleRate* algorithm based on this observation. Instead of using loss thresholds and merely increasing or decreasing the bit rate to the next higher or next lower level like earlier algorithms do, *SampleRate* explicitly calculates throughput for different rates under prevailing network conditions and selects the rate that maximizes throughput even though this rate may not be the next highest or lowest relative to the current rate. *SampleRate* calculates throughput at rates other than the current rate by sending 10% of packets at other rates. The rates thus sampled are chosen intelligently, *i.e.*, only those rates, based on modulation and encoding, that can potentially increase throughput relative to the current rate are sampled.

2.3.1.2 SNR-based Rate Adaptation Algorithms

Holland *et al.* in [56] present the *Receiver-Based AutoRate Protocol (RBAR)*, an SNR-based rate-adaptation algorithm where the receiver picks the bit rate. The receiver picks the highest rate that keeps BER below a certain threshold. Since the relationship between SNR and BER is environment-specific, *RBAR* uses an analytical model to estimate it. The motivation behind receiver-based rate selection is that the packet's SNR information upon reception is available at the receiver and not at the sender (which only knows whether the packet was successfully transmitted or lost), and the finer-grained SNR information facilitates better rate selection and allows the rate to be changed at per-packet granularity if necessary. The disadvantage of *RBAR* is that it assumes the use of RTS/CTS and changes to RTS/CTS headers to allow the receiver to communicate the rate to the sender.

Pavon *et al.* in [99] present a more practical SNR/RSSI-based approach based on the assumption of link reciprocity on the sender-receiver and receiver-sender directions. The sender observes all the receiver's packet transmissions regardless of their destination and records the RSSI and the number of retransmissions. The sender uses the relationship between the RSSI and the number of transmissions to pick a rate that keeps losses below a certain threshold, assuming that the behavior of its transmissions to the receiver will be similar due to link reciprocity. While this algorithm does not assume the use of RTS/CTS or changes to 802.11 headers, it assumes the presence of enough transmissions from the receiver to estimate accurately the relationship between RSSI and loss rate. Judd *et al.* in [65] present *CHARM*, another algorithm that uses link reciprocity and observations of the receiver's transmissions to allow RSSI-based rate control by the sender without the use of RTS/CTS or changes to 802.11 headers for both stationary and vehicular nodes. Zhang *et al.* in [139] present an SNR-based rate adaptation algorithm that filters out the effect of interference on SNR so that rate decreases occur only due to increase in channel noise and not due to increase in interference.

2.3.1.3 Hybrid Rate Adaptation Algorithms

Sadeghi *et al.* in [107] present *Opportunistic Auto Rate (OAR)*, a medium access and reservation scheme designed to increase network throughput in conjunction with SNR-based rate adaptation algorithms. It is based on the observation that the RF environment changes on the granularity of multiple packet transmissions rather than single packet transmissions, so when an SNR-based algorithm decides to increase the rate, it should reserve the medium for multiple packet transmissions to make the most of good channel conditions. However, *OAR* assumes the presence of RTS/CTS so it is not implemented in practice.

Haratcherev *et al.* in [53] present a hybrid rate adaptation algorithm designed specifically for real-time streaming multimedia content. The algorithm has two operating modes, one for stable channel conditions, and one for volatile (*i.e.*, rapidly changing) channel conditions. Under stable conditions, a frame-based rate adaptation algorithm is deployed to maximize throughput, but under volatile conditions an SNR-based approach that can adapt the rate at per-packet granularity is deployed to lower the rate rapidly if needed to meet real-time constraints.

Collision-Aware Rate Adaptation (CARA) [70] is a frame-based algorithm that uses RTS/CTS. It improves on *ARF* by including sender-side RTS/CTS-based heuristics to determine whether losses are due to channel noise or interference, and decreases the rate only in response to increase in the former. Hence, it incurs the RTS/CTS overhead, but because it is a sender-side-only algorithm, it does not require changes to 802.11 headers. The *Robust Rate Adaptation Algorithm (RRAA)* presented in [132] uses short term frame-based performance averages, rather than the longer term averages used by other *ARF*-based algorithms, to increase rate adaptation agility. It also uses an RTS/CTS-based heuristic to distinguish between noise and collision losses. The *History-Aware Robust Rate Adaptation Algorithm (HA-RRAA)* [101] improves on *RRAA* by including recent channel dynamics history in rate change decisions. Wang *et al.* in [130] attempt to solve the problem of rate adaptation algorithms mistaking interference for channel noise using a mathematical model of packet transmission time. Xu *et al.* in [134] propose optimizations that allow algorithms such as *SampleRate* [24] and *CHARM* [65] to work well in vehicular environments.

Vutukuru *et. al* in [129] present a cross-layer rate adaptation scheme that attempts to combine the best of all approaches, *i.e.*, the agility of SNR-based methods, the zero traffic overhead of frame-based methods, and the ability to distinguish between losses due to collisions and channel noise. However, this approach requires changing the interface between the device driver and the wireless card to provide more physical layer information to the driver, so it is not practical for current commodity hardware.

2.3.2 Trace-based Network Application Identification

Our work identifies the 802.11 rate adaptation algorithms deployed in a network based on the analysis of passively acquired network packet traces. There is a large body of work in the literature that focuses on identifying the applications (web, peer-to-peer, bulk transfers, interactive, transactional, etc.) deployed in a network. While our work deals with identifying rate adaptation algorithms at the MAC layer, and the application classification work deals with identifying phenomena at the application layer, both are similar in that they are *in the dark* approaches, *i.e.*, both approaches are based on the recognition of implicit characteristics such as the statistical or time series patterns in the traces rather than explicit characteristics such as port numbers or packet payload contents. We outline *in the dark* application classification approaches in this section.

There are two main motivations for application classification.[2] The first is to allow network administrators to track how their networks are being used, and to facilitate tasks such as resource allocation and blocking of undesirable applications. The second is to enable traffic to be assigned appropriate priorities based on the application's quality of service (QoS) needs. Applications have been traditionally identified based on the ports they use, but over the last decade this technique has become ineffective due to the use of non-standard ports by several classes of applications such as P2P and due to applications using HTTP as the underlying protocol. And, as encryption of network data has become the norm, application classification techniques no longer have access to application data. Techniques for *in the dark* application classification have been developed in response to the prevalance of non-standard port numbers and packet payload encryption.

There are two major approaches to *in the dark* application classification. The first is flow-level classification based on various moments of packet trace features such as packet size, packet count, packet interarrival time and data volume. The second is based on considering the *social* signature of an application, *i.e.*, by analyzing how many other hosts and ports it communicates with and whether it is a producer or consumer of data.

Techniques that identify applications based on flow-level classification have two key components. The first is the selection of a set of flow level features, such as packet counts, packet sizes, and packet interarrival times, that are believed to be most useful in distinguishing between applications. The second is the selection of classification techniques for calculating the degree of similarity between feature vectors to infer the application class of a flow. Roughan *et al.* [106] use k-Nearest-Neighbor and Linear Discriminant Analysis for classification. Since these classification techniques work well only for a small number of features, they experiment with small subsets of features such as various moments of packet size, flow duration, data volume, number of packets, advertised windows and packet interarrival times. They find that the most effective three features for classification are average packet size, flow duration, and interarrival time variability. McGregor *et al.* [85] use Expectation Maximization (EM) on minima, maxima and quartiles of packet sizes, interarrival times, byte counts, connection durations, numbers of transitions between bulk and transaction modes and the amount of idle time spent in bulk and transaction mode to classify flows. Erman [41] use K-Means, a partition-based algorithm, and DBSCAN, a density-based algorithm, on features similar to those used in [85], and find that both these algorithms are more computationally efficient than EM. Moore *et al.* [89] use Naïve

[2] A third motivation is distinguishing between legitimate and malicious application traffic. In this section, we limit ourselves to the classification of legitimate applications. The identification of malicious traffic is discussed in Chapter 3 Section 3.7.

Bayes and Naïve Bayer Kernel estimation for flow classification – the latter method, unlike the former, does not require feature values to be normally distributed. They begin with a set of 248 features, and use heuristics to eliminate irrelevant or redundant features to improve classification accuracy. Wright *et al.* [133] use Hidden Markov Models to identify applications. Crotti *et al.* [35] use normalized thresholds on the Probability Density Functions (PDFs) of flow properties such as packet size, packet direction, packet arrival order and interarrival time for application classification. Turkett *et al.* [64] use Support Vector Machines for application classification. In addition to using conventional features such as the number of different TCP packet types in the flow, interarrival times and packet sizes, they use features generated from the flow's *spectrum kernel* [78], which is a kernel (Chapter 3, Section 3.5) particularly suited for SVM-based classification of strings.

All the application classification approaches discussed so far classify a flow after its completion. Bernaille *et al.* [21, 20] present an *online early application identification* approach. Online early application idenfication is important because it facilitates appropriate handling of the flow in a timely manner, *e.g.*, by giving it the resources needed for QoS support or blocking it if it is a prohibited application. Their approach classifies an application based on the size and direction of the first P packets of a flow. The classification techniques they use include K-Means, Gaussian Mixture Models, and Spectral Clustering on Hidden Markov Models. Sena *et al.* [111] also use the size and direction of the first P packets for early application classification, but use SVMs as the classification technique.

The studies discussed so far classify applications based on individual flows. Other classification studies consider flow aggregates through backbone routers. Soule *et al.* [117] use histograms based on mean flow bandwidth and Dirichlet Mixture Processes to cluster flows according to the amount of data transferred into *elephants, buffaloes, mice* and *dragonflies*, and identify other distinguishing properties of each cluster. Xu *et al.* [135] use an information theoretic approach to the classification of flow aggregates, using the entropy in the distributions of source and destination IP addresses and ports to cluster flows by applications and also to distinguish between legitimate and malicious traffic.

Karagiannis *et al.* [67] are the first to classify applications based on *social* behavior, *i.e.*, by studying the communication patterns between sets of hosts. They look at communication patterns at three different levels of detail – the social level, which considers the number of hosts a given host communicates with, the functional level, which considers whether a host is a consumer or producer of data or both, and the transport level, which considers the patterns of port number usage – and use the combination of information at each level to come up with a final classification. Allan *et al.* [11] also classify applications based on their social behavior. They use a graph-theoretic classification approach. They construct graphs based on host communication patterns, and search for *motifs*, which are frequently occuring subgraphs, to identify similar applications. The main problem associated with this approach is that it scales poorly as the size of subgraphs increases. Finally, Szabo *et al.* [121] use a combination of the social approach [67] and the information theoretic approach [135] for application classification, and find that the combination improves the classification confidence and reduces the amount of unclassified traffic.

2.3.3 Our Contributions Relative to Prior Work

To our knowledge, there have been no other efforts to identify the rate adaptation algorithms deployed in a wireless network. Our work is complementary to work on the design of 802.11 rate adaptation algorithms. In this section, we discuss why there was a need for an effort such as ours for 802.11 rate adaptation algorithm identification, *i.e.*, why some technique from the large body of work on *in the dark* application identification presented in Section 2.3.2 cannot be used directly or with trivial adaptations for 802.11 rate adaptation algorithm identification.

802.11 rate adaptation algorithms are a MAC layer feature, and algorithm behavior depends only on the unidirectional loss rate on the link under consideration. As a result, the type of information available for rate adaptation algorithm identification is significantly different from what is available for application identification. Specifically, methods based on the social behavior of hosts and applications, such as [67, 11] cannot be used because rate adaptation algorithm behavior is specific to one link. Methods that use the size and direction of packets [21, 20, 111] cannot be used either because rate adaptation algorithms (a) deal with unidirectional links and (b) packet payload size is a construct of the transport and application layer, and rate adaptation algorithm behavior does not depend on packet size.

Adapted versions of flow-feature based classification techniques such as [106, 85, 41, 89, 133, 35] can potentially be used for 802.11 rate adaptation algorithm identification. However, our work is distinguished from these methods by (a) our choice of SVM for classification and (b) our feature selection.

The use of SVM as the classification method sets our work apart from the above-mentioned flow-feature based classification efforts. SVM is a much more powerful and robust classifier compared to techniques such as k-Nearest-Neighbor, Linear Discriminant Analysis, Expectation Maximization and Naïve Bayes because (a) it does not impose any restrictions on the properties of input features such as requiring them to be independent or normally distributed, and (b) it can easily and efficiently handle large numbers of input features. Both these properties of SVM are essential for rate adaptation algorithm classification because the input features are neither independent nor normally distributed.

Our careful feature selection also sets our work apart from the above-mentioned flow-feature based classification efforts. Most of these efforts are based on the use of a small number of key features such as packet size and packet interarrival time because (a) a small number of features are sufficient for application classification, and/or (b) the classification methods used can handle only a small number (2–10) of features. However, as Chapter 6 Section 6.4 shows, a large number (approximately 4000) of carefully selected features are necessary for high classification accuracy for rate adaptation algorithms.

Turkett *et al.* [64] use SVMs for application classification so they are not restricted in terms of the number of input features they can use. In addition to using conventional features such as packet sizes and interarrival times, they generate features using a string kernel called the *spectrum representation*. In contrast, our method is able to generate high classification accuracy using only a linear kernel. Unlike [64], our input features do not undergo any complicated mapping before classification, so with our technique it is possible to explain classification accuracy in terms of the choice of features (Chapter 6 Section 6.4). Hence careful feature selection and the ability to provide an intuitive explanation of the relationship between feature selection and accuracy in spite of the presence of a large number features are key contribution of our work and set it apart from prior work.

Chapter 3

Support Vector Machines

Computer networks are becoming more complex with time due to the increased sophistication of deployed technologies, protocols and algorithms and the growth in network scale in terms of the number of users and the volume of data transferred. As the number of variables affecting network performance increases, performance analysis using conventional formula-based and history-based techniques becomes challenging or even impossible. Machine learning techniques, which are designed to handle complex relationships between large numbers of variables, have thus been gaining popularity as an alternative to conventional network analysis methods. In this dissertation, we make a case that Support Vector Machines (SVMs), a state-of-the-art machine learning technique, offers an effective framework for network performance analysis in the face of growing network complexity.

In this chapter, we present an overview of SVMs. Our objective is to provide the reader with a catalog of the main ideas behind SVMs. The explanations and derivations presented in this chapter, borrowed from a wide variety of sources [55, 19, 25, 90, 22, 88], are based on geometric arguments and have been selected for their simplicity and clarity. Mathematically rigorous treatment of SVMs can be found at [126, 109, 113].

3.1 Overview

SVMs are one of the most accurate, if not the most accurate, ”out-of-the-box” supervised learning methods available at present. SVMs are able to capture complex relationships between large numbers of variables while remaining computationally efficient. They have the advantage of being firmly grounded in statistical learning theory [126] while at the same time delivering high accuracy in practice for a diverse set of applications such as text categorization [62], face identification [93] and bioinformatics [110].

Three concepts are fundamental to the understanding of SVMs: *maximum margin linear classifiers*, *Lagrangian duality*, and *kernels*.

In the training phase, SVMs take as input labeled data points from different *classes*, where classes correspond to categories from the problem domain, and construct a *linear classifier* to separate the classes. In the test phase, the SVM has to assign a class label to unlabeled data points, and the goal is to maximize the classification accuracy.

SVMs are based on the idea, formalized in statistical learning theory [126] that given a finite amount of training data, the best accuracy on test data is achieved by finding the right balance between accurate classification of the training data and the *capacity* of the machine. The capacity of a machine is a measure of its expressiveness, *i.e.*, its ability to learn complex functions. The greater the capacity of a machine, the more effectively it is able to learn a particular training set. It is important that the machine have sufficient capacity to classify the training data with reasonable accuracy, because if the training data cannot be classified accurately, it is unlikely that test data will be classified accurately. On the other hand, a machine with high capacity can *overfit* the training data, *i.e.*, learn a function that is so specific to the quirks and noise in the training data, that it will be unable to generalize to test data. Statistical learning theory shows that the *maximum margin linear classifier*, *i.e.*, the linear classifier or hyperplane that separates the data such that it maximizes the distance between itself and the points closest to it in the training set, strikes the optimal balance between accurate training set classification and capacity. The computation of the maximum margin linear classifier is a constrained optimization problem with a quadratic objective and linear constraints, so it can be solved using quadratic programming.

Instead of solving the SVM optimization problem in its original form, however, it is converted into its *Lagrangian dual* form. There are two advantages of this switch. The first is that the optimization constraints are replaced by constraints on the Lagrange multipliers, which can be solved more efficiently. The second is that input data appears in the Lagrangian dual only in the form of dot products. This property turns out to be crucial for the extension of SVMs for data that cannot be classified accurately by a linear boundary and requires non-linear boundaries.

The conventional solution for handling classification problems requiring non-linear boundaries is to map the data into a higher dimensional space where the data can be separated by a linear boundary. The problem with this approach is that working in high dimensional spaces is computationally expensive, sometimes too expensive to be practical. However, since input data appears in the Lagrangian dual of the SVM optimization problem only in the form of dot products, *kernels* can be used to circumvent the computational intractability of working in high dimensional spaces. Kernels are functions that can approximate the dot product of two points in a higher dimensional space directly, without having to explicitly map the points into the higher dimensional space and then calculating the dot product. Thus kernels give SVMs greater expressive power by enabling the construction of non-linear boundaries, but without incurring the computational cost of working in higher dimensional spaces. The combination of Lagrangian duality and kernels means that SVMs can construct complex, non-linear classifiers while retaining the form of a linear learning method which has a solid theoretical foundation and well-understood properties.

SVMs are based on a convex optimization problem, so unlike methods such as neural nets and decision trees, they do not suffer from the problem of local minima. The solution to the SVM optimization problem depends only on the points closest to the margin, *i.e.*, the points in the input that are the most difficult to classify accurately, rather than all the points in the input. The fact that only a small number of input data points determine the solution to the SVM is known as the *sparseness* property of SVMs, and allows SVMs to be applied to tasks such as text categorization and bioinformatics that involve large amounts of data.

3.2 Linear Separation with Hyperplanes

We begin our overview of SVMs by introducing the concept of linear classifiers. SVMs are a type of a linear two-class classifier, *i.e.*, they are able to separate two classes using a decision boundary that is linear in the input.

Suppose that we have a set of samples that belong to two different classes, squares and circles, as illustrated in Figure 3.1a. Let the class label for squares be $y = +1$ (positive samples) and that for circles be $y = -1$ (negative samples). Let $\mathbf{x}_d \in \mathbb{R}^d$ be a vector of dimension d, where $\mathbf{x}$ is called the input vector and each field in $\mathbf{x}$ is a feature or attribute that we expect will be useful for distinguishing between the two classes (squares or circles). The notation $\mathbf{x}_i$ denotes the i^{th} vector or example in a dataset of N vectors $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. y_i is the label associated with $\mathbf{x}_i$.

We want to devise a method to separate the two classes. We assume for now that the two classes are linearly separable, and later extend to the case where the classes are non-separable. A large number of possible hyperplanes can separate the two classes, three of which are illustrated in Figure 3.1b. We are now faced with the task of selecting the best hyperplane to separate the classes. We would like to find a hyperplane that performs well not just on the training examples, but also minimizes *generalization* error, *i.e.*, misclassification error on test data. For this, we introduce the concept of the *margin* of a linear classifier. The margin of a linear classifier is the width by which the classifier boundary can be increased before it touches a data point, *i.e.*, the distance of the closest example to the decision boundary (Figure 3.1c). Results from statistical learning theory show that the maximum margin hyperplane, *i.e.*, the hyperplane with the widest margin, minimizes generalization error.

Statistical learning theory [126] deals with proving bounds on generalization error. It has shown that generalization error is minimized by maximizing the classification accuracy for a finite-sized training set while at the same time minimizing the *capacity* of the linear classifier. Capacity is a measure of the complexity or expressive power of classification functions. The higher the capacity of a classification function, the greater the complexity of the boundary it can construct to separate training data. To effectively separate the training data, the classification function has to have some minimal capacity. However, as the capacity of the classification function increases, the risk of *overfitting* the training data increases, which in turn increases the risk of misclassifying test data. Hence what is required is a classifier with the correct balance between maximizing accuracy on training data and minimizing capacity. The maximum margin hyperplane is the classifier that achieves this balance.

An intuitive explanation for designating the maximum margin hyperplane as the best separating hyperplane, presented in [90], is the following. Consider Figure 3.1d, which illustrates three points A, B and C in the input space belonging to the class $y = +1$. A is very far from the decision boundary, so if we are asked to make a prediction for the value of y at A, we can be quite confident that $y = +1$ there. However, C is very close to the decision boundary, so we are significantly less confident that it belongs to the class $y = +1$, because a small shift to the decision boundary would have changed the classification to $y = -1$. B lies between A and C, so our confidence in its classification as $y = +1$ is somewhere in between as well. We would like to be both *correct* and *confident* in our classifications, and an increase in distance from the decision boundary makes us more confident of the correctness of a classification. To maximize the confidence in the classifications, we need to maximize the margin, *i.e.*, to find the hyperplane with the largest distance from the input samples. The determination of the maximum margin depends only on the points in each class closest to the decision boundary. These points are known as the *support vectors*, and for our example are circled in Figure 3.1e. We describe how to find the maximum margin classifier in the remainder of this section.

A key concept required for defining a linear classifier is the *dot product* or *scalar product* between two vectors $\mathbf{w}^\top \mathbf{x} = \sum_{i=1}^{d} w_i x_i$. A linear classifier is based on a *linear discriminant function* of the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \tag{3.1}$$

The training data is linearly separable, so we classify the input as follows,

$$\begin{aligned} &+1 \text{ if } \mathbf{w}^\top \mathbf{x}_i + b \geq +1 \\ &-1 \text{ if } \mathbf{w}^\top \mathbf{x}_i + b \leq -1 \end{aligned} \tag{3.2}$$

where $\mathbf{x}_i$ is a data point, $\mathbf{w}$ is a vector of weights perpendicular to the hyperplane and determines its orientation, and b is the y-intercept of the hyperplane (Figure 3.1e). The inequalities in (3.2) indicate that the separating hyperplane divides the input data into two half spaces, with the sign of $f(\mathbf{x})$ determining whether a data point belongs to the positive or negative class, *i.e.*, determining the side of the hyperplane on which a point lies.

Our goal is to find the maximum margin hyperplane. For this, consider two additional hyperplanes parallel to a separating hyperplane, such that one touches the support vectors of the positive class and the other the support vectors of the negative class. We refer to the former as the *Plus* plane, and the latter as the *Minus* plane, Figure 3.1e[1]. We have to find the maximum margin width, i.e., the distance between the *Plus* and *Minus* planes, in terms of $\mathbf{w}$ and b. To do this, we scale the discriminant function (3.1) such that equality holds for both inequalities in (3.2) for the points on the *Plus* and *Minus* planes,

$$\begin{aligned} \textit{Plus} \text{ plane: } &\{\mathbf{x} \colon \mathbf{w}^\top \mathbf{x} + b = +1\} \\ \textit{Minus} \text{ plane: } &\{\mathbf{x} \colon \mathbf{w}^\top \mathbf{x} + b = -1\} \end{aligned}$$

Let $\mathbf{x}^-$ be any point on the *Minus* plane, and $\mathbf{x}^+$ be the closest point to $\mathbf{x}^-$ on the *Plus plane*. Then $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$ for some value of λ, because $\mathbf{w}$ is perpendicular to both the *Plus* and *Minus* plane. It follows from the definitions of

[1]This explanation is borrowed from [88]

$\mathbf{x}^+$ and $\mathbf{x}^-$ that

$$\mathbf{w}^\top\mathbf{x}^+ + b = +1 \tag{3.3}$$

$$\mathbf{w}^\top\mathbf{x}^- + b = -1 \tag{3.4}$$

$$\mathbf{x}^+ = \mathbf{x}^- + \lambda\mathbf{w} \tag{3.5}$$

The margin width, M, is the norm of the difference in the position vectors of $\mathbf{x}^+$ and $\mathbf{x}^-$,

$$\|\mathbf{x}^+ - \mathbf{x}^-\| = M \tag{3.6}$$

Substituting (3.5) for $\mathbf{x}^+$ in (3.3) and solving for λ, we get:

$$\begin{aligned}
&\mathbf{w}^\top(\mathbf{x}^- + \lambda\mathbf{w}) + b = 1\\
\Rightarrow&\mathbf{w}^\top\mathbf{x}^- + b + \lambda\mathbf{w}^\top\mathbf{w} = 1\\
\Rightarrow&-1 + \lambda\mathbf{w}^\top\mathbf{w} = 1\\
\Rightarrow&\lambda = \frac{2}{\mathbf{w}^\top\mathbf{w}}
\end{aligned} \tag{3.7}$$

Hence M, the margin width, is:

$$M = \|\mathbf{x}^+ - \mathbf{x}^-\| = \|\lambda\mathbf{w}\| = \lambda\|\mathbf{w}\| = \lambda\sqrt{\mathbf{w}^\top\mathbf{w}} = \frac{2\sqrt{\mathbf{w}^\top\mathbf{w}}}{\mathbf{w}^\top\mathbf{w}} = \frac{2}{\sqrt{\mathbf{w}^\top\mathbf{w}}} \tag{3.8}$$

To maximize the margin M, we need to maximize $\frac{2}{\sqrt{\mathbf{w}^\top\mathbf{w}}}$, *i.e.*, to minimize $(\mathbf{w}^\top\mathbf{w})$. Hence, for linear SVMs, we have to solve the following optimization problem:

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.9}$$

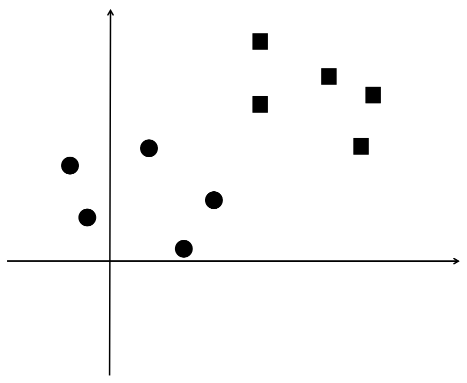
subject to the constraints

$$\begin{aligned}
\mathbf{w}^\top\mathbf{x}_i + b \geq +1 \text{ for } y_i = +1\\
\mathbf{w}^\top\mathbf{x}_i + b \leq -1 \text{ for } y_i = -1
\end{aligned}$$

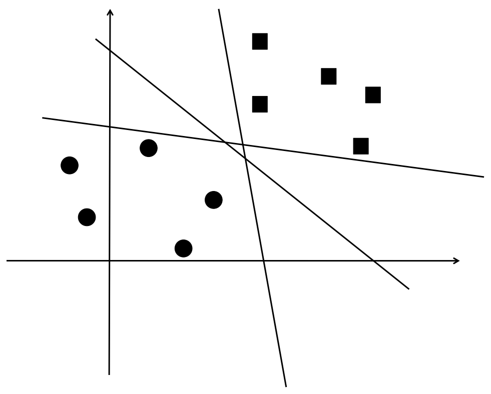
The above constraints can be combined into one set of inequalities:

$$y_i(\mathbf{w}^\top\mathbf{x}_i + b) - 1 \geq 0 \quad i = 1, \dots, N \tag{3.10}$$

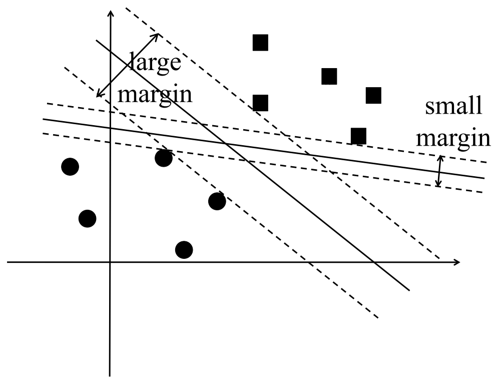
where N is the number of training samples.



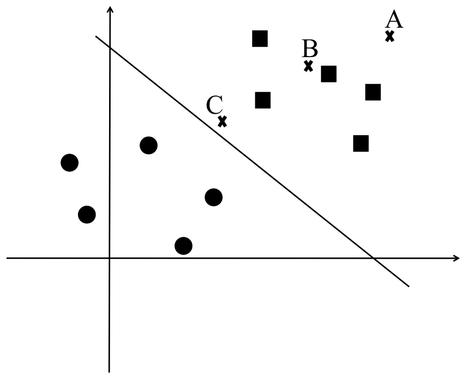
(a) Two classes to be separated



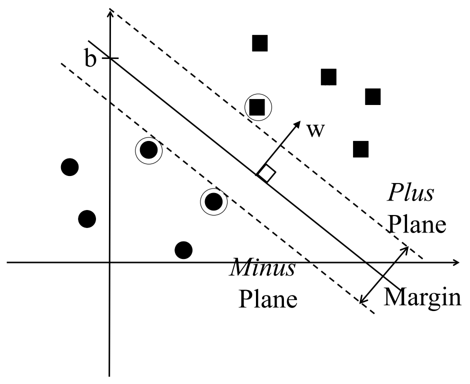
(b) Many possible separating hyperplanes



(c) Large and small margins



(d) Reason for selecting large margins



(e) Maximum margin hyperplane

Figure 3.1: Geometric representation of the linear Support Vector Machines optimization problem

3.3 The Dual Formulation of the SVM Optimization Problem

The above constrained optimization problem has a convex quadratic objective and linear constraints, so it can be solved using available commercial *quadratic programming (QP)* packages. However, we will transform the problem with the use of Lagrange multipliers into a form that is more efficiently solvable by quadratic programs. This form also allows the linear SVMs to be modified easily to non-linear SVMs, as discussed in Section 3.5.

Lagrange multipliers incorporate the constraints into the expression to be minimized or maximized. For a function $f(\mathbf{x})$ and m constraints $g_i(\mathbf{x}) = 0$, the Lagrangian L is defined as

$$L(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_{i=1}^{m} \alpha_i g_i(\mathbf{x})$$

The α_i's are the Lagrange multipliers. The values of the $\mathbf{x}_i$'s and α_i's are determined by setting the partial derivatives to zero

$$\frac{\partial L}{\partial x_i} = 0 \text{ for } 1 \leq i \leq n \qquad (n \text{ equations})$$

$$\frac{\partial L}{\partial \alpha_i} = 0 \text{ for } 1 \leq i \leq m \qquad (m \text{ equations})$$

and solving the resulting system of $m + n$ equations.

For positive constraints of the form $c_i \geq 0$ such as (3.10), the rule is that the constraint equations are multiplied by positive Lagrange multipliers and subtracted from the objective function to form the Lagrangian. Hence the Lagrangian for the linear SVM optimization problem (3.9) with constraints (3.10) is

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w}^\top \mathbf{x}_i) - 1) \tag{3.11}$$

There are positive Lagrange multipliers α_i's, $i = 1, \ldots, N$ for each of the inequality constraints (3.10). We denote the Langragian by L_P, where the subscript P means that this is the *primal* formulation of the optimization problem. The Langragian can be solved by setting the partial derivatives of L_P to zero

$$\frac{\partial L_P}{\partial w} = 0 \quad \Rightarrow \mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{3.12}$$

$$\frac{\partial L_P}{\partial b} = 0 \quad \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.13}$$

and by incorporating the Lagrangian constraints

$$\alpha_i \geq 0 \tag{3.14}$$

Solving for $\mathbf{w}$, b and all α_i based on (3.12), (3.13) and (3.14) is a complicated task, so we turn to an alternate formulation of the optimization problem known as the *Wolfe dual*, L_D. The Wolfe dual has the property that its maximum subject to some constraints C_D occurs for the same values of $\mathbf{w}$, b and α, as the minimum of L_P subject to some constraints C_P. (3.12) and (3.13) are equality constraints in the dual formulation, so we substitute them into (3.11) to give

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{3.15}$$

The linearly separable SVM optimization problem is now

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{3.16}$$

with constraints

$$\alpha_i \geq 0 \tag{3.17}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{3.18}$$

The dual problem has constraints that are more manageable than that of the primal problem, and hence can be solved more efficiently.

The Karush-Kuhn-Tucker (KKT) conditions, listed below, hold for the solutions of all support vector machines because the constraints are always linear:[2]

$$\frac{\partial L_P}{\partial w_v} = w_v - \sum_{i=1}^{N} \alpha_i y_i x_{iv} = 0 \qquad v = 1, \ldots, d \tag{3.19}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i \tag{3.20}$$

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \geq 0 \qquad i = 1, \ldots, N \tag{3.21}$$

$$\alpha_i \geq 0 \qquad i = 1, \ldots, N \tag{3.22}$$

$$\alpha_i(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1) = 0 \qquad i = 1, \ldots, N \tag{3.23}$$

According to (3.23), the KKT *dual complementary* condition, for all instances where $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \neq 1$, *i.e.*, when a point is not on either the *Plus* plane or the *Minus* plane, it must hold that $\alpha_i = 0$. Only for points that lie on either the *Plus* or the *Minus* plane is $\alpha_i > 0$. Recall that these points are the *support vectors* discussed in Section 3.2 and illustrated in Figure 3.1e. The solution to the SVM, *i.e.*, the maximum margin hyperplane, thus depends only on the support vectors. In other words, if all other points were removed, or moved around but such that they do not cross the

[2]In the listing of the KKT conditions, i runs from 1 to the number of training points, and v from 1 to the dimension of the data.

Plus or the *Minus* plane, the maximum margin hyperplane would not change. The fact that only the support vectors, which are generally a small fraction of the input samples, affect the solution to the SVM is known as the *sparseness property* of SVMs.

Once the α_i values have been determined, we can compute $\mathbf{w}$ as follows:

$$\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i y_i \mathbf{x}_i$$

where N_s is the number of support vectors. b is obtained by inserting the support vectors into the KKT dual complementary condition $\alpha_i(y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1) = 0$ and solving for b. For robustness, the average value of b over all support vectors should be used.

One of the benefits of the dual formulation is that the input data appears in the algorithms only in the form of dot products. As we will discuss in Section 3.5, this allows for easy generalization to non-linear SVMs, and for efficient computation in very high dimensional spaces.

So far, we have limited our discussion to binary SVMs, *i.e.*, SVMs that can distinguish between only two different input classes. Binary SVMs can be adapted to distinguish between multiple input classes. Such SVMs are known as *multi-class* SVMs. The simplest way to construct a multi-class SVM from a binary SVM is the *one against all* approach. In this approach, if there are N classes, N different machines are computed, with each machine separating one class from all the other $N - 1$ classes. To classify a new input, a prediction is made with each machine, and the predicted class is the one whose machine puts the input the farthest from the margin into the positive class. A number of attempts have also been made to handle multiclass input in a single step [34, 131, 102].

3.4 Soft Margin Classifier

So far, we have looked at the case where the data can be separated by a linear boundary. We now look at how SVMs handle the case where the data cannot be separated by a linear boundary. An example is presented in Figure 3.2a, with the gray circles and squares indicating the samples that will be misclassified by a linear boundary. This often occurs due to noise in real data. In such a case, it is better to misclassify a few points and find a robust wide margin rather than finding a narrow linear margin or a nonlinear boundary using kernels (Section 3.5) because the last two result in the classifier *overfitting* the training data and hence generalizing poorly to test data.

To handle non-separable data, *slack variables* ξ_i are introduced to relax the constraint (3.10) by allowing a point to be a small distance ξ on the wrong side of the hyperplane:

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad i = 1, \ldots, N \tag{3.24}$$

Such a hyperplane, called a *soft margin classifier*, is illustrated in Figure 3.2b. Constraint (3.24) allows the trivial solution where extremely large slacks will allow any line to ”separate” the data. To prevent this, a constant term C is added to the objective function (3.9):

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{3.25}$$

The value of the constant $C > 0$ determines the tradeoff between maximizing the margin and minimizing the number of misclassified samples. The Wolfe dual optimization problem is

$$\max_{\alpha} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\mathbf{x}_j \tag{3.26}$$

subject to

$$\alpha_i \geq C \tag{3.27}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{3.28}$$

$$\xi_i \geq 0 \tag{3.29}$$

The solution is again given by

$$\mathbf{w} = \sum_{i=1}^{N_s}\alpha_i y_i \mathbf{x}_i$$

where N_s is the number of support vectors. Note that neither the ξ_i’s nor their Lagrange multipliers appear in the dual formulation, and the only difference from the separable case is that the upper bound on the value of the α_i’s is C instead of 1. The value of b can be calculated by considering the primal formulation of the problem and the KKT dual complementary conditions – a complete derivation can be found in [25].

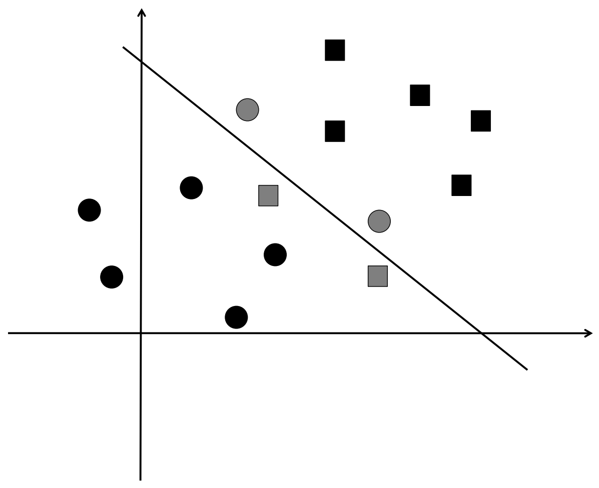
3.5 Nonlinear Support Vector Machines: Kernels

We now consider the case where a linear decision function cannot separate the data. This problem can be solved by mapping the data using a function Φ into a higher, possibly infinite, dimensional space in which the data is linearly separable. The data points in the SVM optimization problem (3.16) are replaced by their mapping under Φ

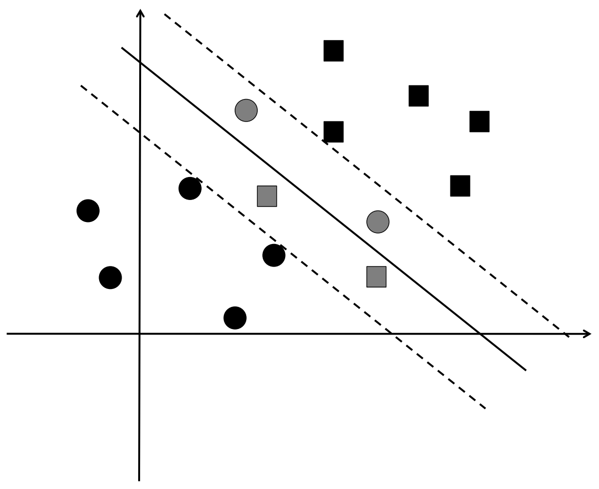
$$\max_{\alpha}\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) \tag{3.30}$$

and the remainder of the analysis is carried out as before.

The problem here is that mapping into a higher dimensional space and then computing the dot product may be so computationally expensive as to be infeasible. The solution to this problem is based on the observation that the input



(a) Linearly inseparable classes



(b) Soft margin classifier

Figure 3.2: Soft-margin classifiers for linearly inseparable data

data appears in the optimization problems (3.16) and (3.30) only in the form of dot products. There exist *kernels* K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi x_i \Phi x_j$$

i.e., the kernel is a function that computes the dot product of two points very efficiently, without having to map the points into higher dimensional space or without even knowing explicitly what the mapping is. SVMs use kernels to eliminate the problem of computing with vectors in very high dimensional space. With the use of kernels, the SVM optimization problem (3.16) becomes

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.31}$$

This leads us to the question of how to find kernels. Mercer's theorem states that any symmetric positive semidefinite matrix is a kernel in some space. Two kernels that perform very well for SVMs are the polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \top \mathbf{x}_j)^d$, which maps the input data into a finite dimensional space, and the Radial Basis Function (RBF) kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, which maps the input data into an infinite dimensional space.

Intuitively, the value of $K(\mathbf{x}_i, \mathbf{x}_j)$ is a measure of the similarity between the points $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$. If the points are similar, the dot product, *i.e.*, the value of $K(\mathbf{x}_i, \mathbf{x}_j)$ is large, and if the points are dissimilar or orthogonal, it is very small or zero. If the mapping Φ is into a space with many irrelevant features, the kernel matrix becomes diagonal and without any clusters or structure, and the resulting margin will not separate the two classes very effectively. A pitfall at the other end of the spectrum is the introduction of an excessive amount of structure by mapping the input data into an overly complicated space, which results in *overfitting*, *i.e.*, the boundary becoming so specific to the training data that it will not generalize well to test samples. This problem is known as *the curse of dimensionality*. In both cases, the outcome will be poor classification accuracy on test data. To avoid both these problems, kernels that measure the similarity between points accurately are needed. An understanding of what makes data points similar or dissimilar generally depends on the domain of the data, so a good understanding of the input data and the problem domain is needed to select a good kernel.

3.6 Support Vector Regression

We now describe how SVMs are used for regression. We present the case where the candidate function is linear in the input. Non-linear regression functions can be handled using kernels as discussed in Section 3.5.

For Support Vector Regression (SVR), the input data is in the form $\{(\mathbf{x}_1, y_1), \ldots, \mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is an input vector of dimension d and y_i is the measured value of the function at $\mathbf{x}_i$. The function to be deduced is approximated by a linear function of the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \tag{3.32}$$

and similar to the classification case we have to determine the value of the weight vector $\mathbf{w}$ and the y-intercept b. The goal is to find a flat $\mathbf{w}$, *i.e.*, a small $\mathbf{w}$. One definition of the smallness of $\mathbf{w}$ is to minimize its norm $\|\mathbf{w}\|^2 = \mathbf{w}^\top \mathbf{w}$, and SVR uses this definition. SVR uses ε-insensitive loss, illustrated in Figure 3.3b, as the constraint for this minimization. ε-insensitive loss means that our goal is to find a function $f(\mathbf{x})$ that deviates at most ε from the measured values y_i. Errors larger than ε are not acceptable, or are penalized, as we will discuss in (3.34). The SVR optimization problem can be written as

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.33}$$

subject to the constraints

$$\begin{gathered} \mathbf{w}^\top \mathbf{x}_i + b - y_i \geq -\varepsilon \\ \mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \varepsilon \end{gathered}$$

Figure 3.3a illustrates the optimization problem geometrically. The solid line through the points corresponds to the case $\mathbf{w}^\top \mathbf{x}_i + b - y_i = 0$. The points within a tube of radius ε around this line correspond to the points that satisfy the above constraints. For real data, it is generally unrealistic to assume that a function f exists that approximates all pairs $(\mathbf{x}_i, y_i)$ within the margin of ε, so we allow some errors by introducing slack variables ξ_i and ξ_i*, with the former indicating overestimation errors and the latter underestimation errors. However, to prevent the trivial solution where very large slack allows every linear function of the form (3.32) to be a solution to the optimization problem, a cost function which scales the slack variables by a constant $C > 0$ is added to the term to be minimized. C determines the tradeoff between minimizing $\|\mathbf{w}\|^2$ and minimizing the amount of allowed error. The SVR optimization problem with slack variables becomes

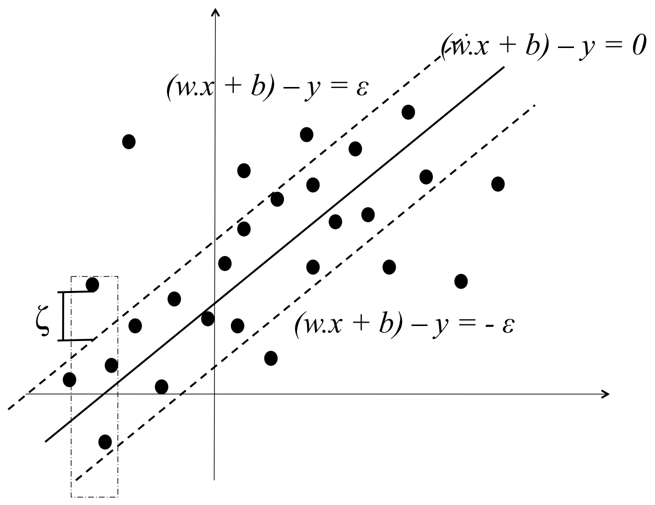
$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i*) \tag{3.34}$$

subject to

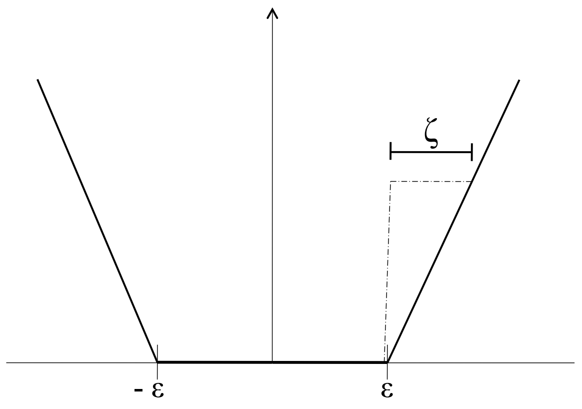
$$\begin{gathered} \mathbf{w}^\top \mathbf{x}_i + b - y_i \geq -\varepsilon - \xi_i* \\ \mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i \\ \xi_i,\ \xi_i* \ \geq 0 \end{gathered}$$

Only the points outside the tube of radius of ε in Figure 3.3a contribute to the cost term in (3.34). The amount of penalty associated with an error is linear in the magnitude of the error, as illustrated by the dotted box in Figure 3.3a and dotted lines in Figure 3.3b.

Similar to SVMs, the SVR optimization problem can be solved more easily in its Lagrangian dual form and using the KKT conditions, complete derivations of which can be found in [113].



(a) Margin width calculation for SVR



(b) Epsilon-insensitive loss for SVR

Figure 3.3: Geometric representation of Support Vector Regression (SVR)

3.7 Applications of SVMs in Computer Networking

In this section, we present examples of the use of SVMs for solving problems related to computer networking. The goal is not to provide an exhaustive listing of the use of SVMs, but rather to provide a flavor of the type of problems most amenable to SVM-based solutions. We present three example problems, (a) network intrusion and anomalous traffic detection, (b) node localization in wireless networks, particularly wireless sensor networks, and (c) link adaptation in MIMO-OFDM wireless networks.

Network intrusion or anomaly detection problems are natural candidates for SVM-based solutions because they require traffic to be classified as either normal or anomalous based on empirical data, *i.e.*, based on labelled examples from packet traces rather than based on some formula or model. Example of studies that use SVMs for intrusion detection are [119, 57]. Sung *et al.* [119] state that SVMs are particularly suited for intrusion detection for a number of reasons. The first is that unlike other learning methods SVMs can handle high dimension feature vectors efficiently because of their sparseness property, (Section 3.3), *i.e.*, the computational cost of finding a classifier depends on the number of support vectors rather than the dimensionality of the feature vectors. The second is that SVMs do not suffer from the problem of overfitting even when the feature vector has high dimensionality. The third is that due to their high computational efficiency SVMs can be used for intrusion detection in real time.

Node localization is the problem of determining the physical location of a node in either a wireless sensor network or an indoor wireless network. GPS-based localization cannot be used in sensor networks because sensor nodes are cheap low-end nodes that are not equipped with GPS. GPS does not work well for indoor environments so it cannot be used for indoor wireless networks. Hence, in both cases, alternative localization techniques are needed. SVM-based techniques determine node location by hierarchically dividing the area under consideration into progressively smaller two-dimensional cells. The classification problem is finding the cell a node belongs to. The key to SVMs' success in node localization, specially for sensor networks (in addition to high accuracy), is the extremely low memory and computation cost imposed on sensor nodes. Examples of studies that apply SVMs to the node localization problem include [125, 71, 42].

SVMs have also been used for link adaptation in multiple-input multiple-output orthogonal frequency division multiplexing (MIMO-OFDM) wireless links. MIMO-OFDM links are extremely attractive because they can dramatically improve wireless throughput without using a broader frequency spectrum. However, there are two challenges in realizing the throughput gains in practice. The first is that link peformance is affected by a very large number of factors. The second is that even if a method (most likely learning-based) can be found to solve the link adaptation problem, its resource requirements in terms of processing power and memory would be too high to allow it to be used in real time. SVMs solve both problems. They can handle complicated non-linear relationships between large numbers of variables, and the sparseness property (Section 3.3) allows the solution to be computed efficiently. Examples of studies that apply SVMs to the link adaptation problem include [137, 37, 138].

Chapter 4

Wireline Throughput Prediction

4.1 Overview

The availability of multiple paths between sources and receivers enabled by content distribution, multi-homing, and overlay or virtual networks suggests the need for the ability to select the *best* path for a particular data transfer. A common starting point for this problem is to define *best* in terms of the throughput that can be achieved over a particular path between two end hosts when transferring a given size file using TCP. In this case, the fundamental challenge is to develop a technique that provides an accurate TCP throughput forecast for arbitrary and possibly highly dynamic end-to-end paths.

There are several difficulties in generating accurate TCP throughput predictions. Prior work on the problem has largely fallen into two categories, those that investigate *formula-based* approaches and those that investigate *history-based* approaches. Formula-based methods, as the name suggests, predict throughput using mathematical expressions that relate a TCP sender's behavior to path and end host properties such as RTT, packet loss rate and receive window size. In this case, different measurement tools can be used to gather the input data that is then plugged into the formula to generate a prediction. However, well-known network dynamics and limited instrumentation access complicate the basic task of gathering timely and accurate path information, and the ever evolving set of TCP implementations means that a corresponding set of formula-based models must be maintained.

History-based TCP throughput prediction methods are conceptually straightforward. They typically use some kind of standard time series forecasting based on throughput measurements derived from prior file transfers that are gathered either passively (*e.g.,* by tapping a link) or actively (*e.g.,* by periodically sending a file). In recent work, He *et al.* [54] present convincing evidence that history-based methods are generally more accurate than formula-based methods. However, [54] also carefully outline the conditions under which history-based prediction can be effective. Furthermore, history-based approaches described to date remain relatively inaccurate and potentially heavyweight processes focused on bulk transfer throughput prediction.

The objective of our work is to develop an accurate, lightweight tool for predicting end-to-end TCP throughput for arbitrary file sizes. To achieve this goal, we investigate the hypothesis that the accuracy of history-based predictors can be improved and their impact on a path reduced by augmenting the predictor with periodic measurements of simple path properties. The specific issues our work addresses include the following:

1. identifying the path properties or combination of path properties that maximize the accuracy of TCP throughput prediction,

2. identifying the minimum set of file sizes required to generate accurate throughput predictors for arbitrary file sizes,

3. designing a predictor that is robust to *level shifts*, *i.e.*, is able to resume accurate predictions as quickly as possible following significant changes in path properties, because [54] has shown that this is a significant challenge for history-based predictors, and

4. including a confidence value with predictions, a metric that has received little attention in prior TCP throughput prediction literature.

To resolve the above-listed issues associated with the problem of TCP throughput prediction, we use Support Vector Regression (SVR), a powerful machine learning technique, described in detail in Chapters 1 and 3.

In Chapter 1 Section 1.1 we discuss the criteria on which any new solution to a network performance analysis problem is evaluated, namely that it has to be at least as accurate as, and ideally more accurate than, existing solutions, and make tradeoffs between accuracy, analysis comprehensiveness, extensibility and maintenance, robustness, agility and practicality that are no worse than, and ideally better than, existing solutions. In the remainder of this section, we outline how our SVR-based TCP throughput predictor improves upon the prior art based on all of the above-listed criteria.

We begin by using laboratory-based experiments to investigate how TCP throughput depends on path properties such as available bandwidth (AB), queuing delays (Q), and packet loss (L). The lab environment enables us to gather highly accurate passive measurements of throughput and all path properties, and develop and test our SVR-based predictor over a range of realistic traffic conditions. Our initial experiments focus on bulk transfers. We compare actual TCP throughput with predictions generated by multiple instances of our SVR-based tool trained with different combinations of path properties. Our results show that for bulk transfers under heavy traffic conditions (90% average utilization on the bottleneck link), the SVR-based predictor is 3 times more accurate than a history-based predictor, with 87% of predictions within 10% of actual throughput in contrast to only 32% within 10% of actual for the history-based predictor. Our initial tests were based entirely on passive measurements which are unavailable in the wide area, so we tested our SVR-based approach using practical but less accurate active measurements of AB, Q, and L. The reduction in accuracy of active versus passive measurements resulted in a corresponding reduction in accuracy of SVR-based throughput predictions for bulk transfers under heavy traffic conditions from 87% to 49% of predictions within 10% of actual. However, even with practical active measurements, the SVR-based predictor outperforms the history-based predictor by a factor of 1.6. Hence the SVR-based approach meets the improved accuracy criterion using both high accuracy laboratory-based passive measurements and practical active measurements.

SVR allows us to experiment with different combinations of path properties as input features for TCP throughput prediction. Using this capability, we find that the path properties that provide the most improvement to the SVR-based predictor are Q and L respectively, and that including AB provides almost no improvement to the predictor. This is good news, because active measurement techniques for Q and L are lightweight while those for AB are heavyweight. We show that SVR-based throughput prediction using active measurements of L and Q results in a factor of 13 lower network probe load compared to prior history-based methods using long-lived TCP transfers as described in [54]. Hence the SVR-based approach succeeds on the practicality criterion for network performance analysis method evaluation.

Prior history-based throughput prediction studies focus only on bulk transfers. Prior formula-based studies handle throughput prediction for a variety of file sizes, but as illustrated by [54], they are quite inaccurate in practice even for bulk transfers. The SVR-based approach can predict throughput accurately for a wide range of file sizes when file size is included in the SVR feature vector. We found that a training set of only three file sizes results in accurate throughput predictions for a wide range of file sizes. The ability to extrapolate from three file sizes in training to a wide range of file sizes in testing demonstrates SVR's ability to capture complex relationships between variables. By effectively predicting throughput for a variety of file sizes based on a training set of only three file sizes, the SVR-based enables analysis comprehensiveness while keeping analysis complexity manageable.

Results from [54] showed that *level shifts* in path conditions cause significant problems for history-based throughput prediction due to the difficulty of distinguishing between genuine changes in network conditions and outliers. We extend the basic SVR predictor with a confidence interval estimator based on an assumption that prediction errors are normally distributed, an assumption that we test in our laboratory experiments. Estimation of confidence intervals is critical for on-line prediction, since retraining can be triggered if measured throughput falls outside a confidence interval computed through previous measurements.

We implement the above capabilities, refined in our laboratory testbed, in a tool called PathPerf, discussed in detail in Section 4.5, that we tested on 18 diverse wide area paths in the RON testbed [13]. We find that PathPerf predicts throughput accurately under a broad range of conditions in real networks. Hence the SVR-based approach meets the robustness criterion for network performance analysis method evaluation.

4.2 Experimental Setup

This section describes the laboratory environment and experimental protocol that we used to evaluate our throughput predictor.

4.2.1 Experimental Environment

The laboratory testbed used in our experiments is shown in Figure 4.1. It consisted of commodity end hosts connected to a dumbbell-like topology of Cisco GSR 12000 routers. Both measurement and background traffic was generated and received by the end hosts. Traffic flowed from the sending hosts on separate paths via Gigabit Ethernet to separate Cisco GSRs (hop B in the figure) where it was forwarded on OC12 (622 Mb/s) links. This configuration was created in order to accommodate a precision passive measurement system, as we describe below. Traffic from the OC12 links was then multiplexed onto a single OC3 (155 Mb/s) link (hop C in the figure) which formed the bottleneck where congestion took place. We used an AdTech SX-14 hardware-based propagation delay emulator on the OC3 link to add 25 milliseconds delay in each direction for all experiments, and configured the bottleneck queue to hold approximately 50 milliseconds of packets. Packets exited the OC3 link via another Cisco GSR 12000 (hop D in the figure) and passed to receiving hosts via Gigabit Ethernet.

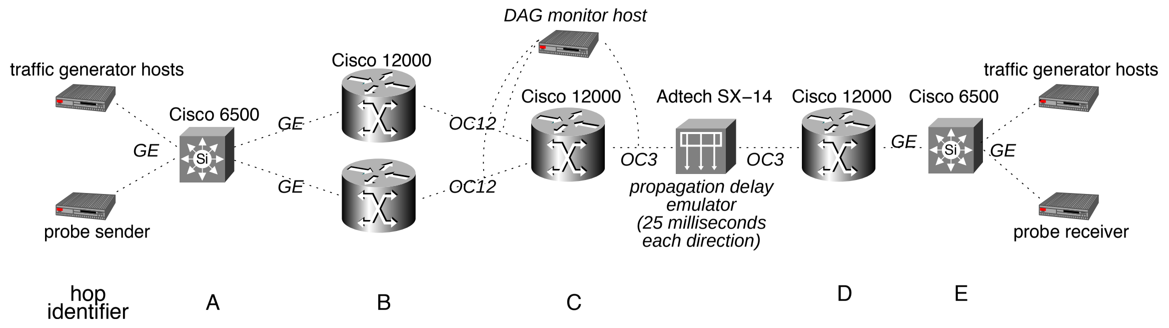


Figure 4.1: Laboratory testbed. Cross traffic flowed across one of two routers at hop B, while probe traffic flowed through the other. Optical splitters connected Endace DAG 3.5 and 3.8 passive packet capture cards to the testbed between hops B and C, and hops C and D. Measurement traffic (file transfers, loss probes, and available bandwidth probes) flowed from left to right. Congestion in the testbed occurred at hop C.

The measurement hosts and traffic generation hosts were identically configured workstations running FreeBSD 5.4. The workstations had 2 GHz Intel Pentium 4 processors with 2 GB of RAM and Intel Pro/1000 network cards. They were also dual-homed, so that all management traffic was on a separate network than depicted in Figure 4.1. We disabled the TCP throughput history caching feature in FreeBSD 5.4, controlled by the variable net.inet.tcp.inflight.enable, to allow TCP throughput to be determined by path properties rather than throughput history.

A key aspect of our testbed was the measurement system used to establish the true path properties for our evaluation. Optical splitters were attached to both the ingress and egress links at hop C and Endace DAG 3.5 and 3.8 passive monitoring cards were used to capture traces of *all* packets entering and leaving the bottleneck node. By comparing packet headers, we were able to identify which packets were lost at the congested output queue during experiments, and accurately measure available bandwidth on the congested link. Furthermore, the fact that the measurements of packets entering and leaving hop C were synchronized at a very fine granularity (*i.e.*, a single microsecond) enabled us to precisely measure queuing delays through the congested router.

4.2.2 Experimental Protocol

We generated background traffic in our testbed by running the Harpoon IP traffic generator [114] between up to four pairs of traffic generation hosts as illustrated in Figure 4.1. Harpoon produced open-loop self-similar traffic using a heavy-tailed file size distribution, mimicking a mix of application traffic such as web and peer-to-peer applications common in today's Internet. Harpoon was configured to produce average offered loads ranging from approximately 60% to 105% on the bottleneck link (the OC3 between hops C and D).

For the measurement traffic hosts, we increased the TCP receive window size to 128 KB (with the window scaling option turned on). In receive window limited transfers, file transfer throughput was approximately 21 Mb/s. That is, if the available bandwidth on the bottleneck link was 21 Mb/s or more, the flow was receive window ($rwnd$) limited, otherwise it was congestion window ($cwnd$) limited. We increased the receive window size because we wanted file transfers to be $cwnd$ limited so we could evaluate our predictor thoroughly under variable throughput conditions. $rwnd$-limited transfers have constant throughput for a given $rwnd$ size, so any reasonable predictor performs well on $rwnd$-limited transfers. For this reason, we do not report any results for $rwnd$-limited flows.

As illustrated in Figure 4.2, measurement traffic in the testbed consisted of file transfers, active measurements of AB using YAZ [116], and active measurements of L and Q using BADABING [115]. We used the most accurate active measurement tools available to us. If tools with greater accuracy become available in the future, their measurements can seamlessly replace YAZ and BADABING measurements in the SVR analysis. We also measured AB, L and Q passively based on packet traces gathered using the instrumentation described in Section 4.2.1. All measurements are aggregates for all flows on the path.

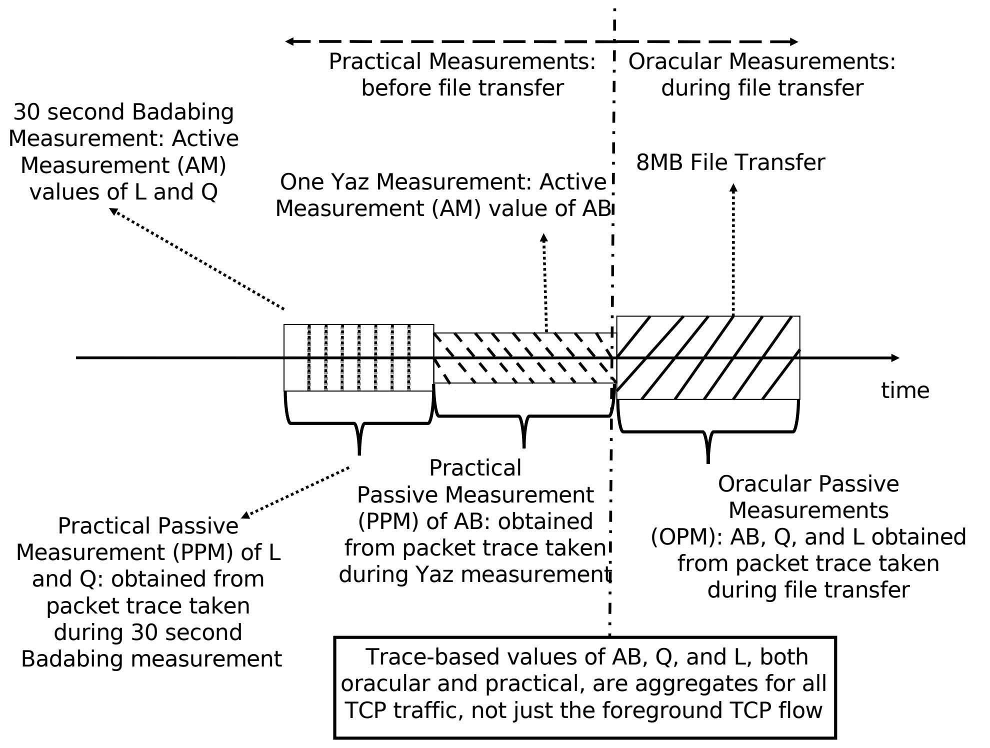

Practical Measurements: before file transfer
Oracular Measurements: during file transfer
30 second Badabing Measurement: Active Measurement (AM) values of L and Q
One Yaz Measurement: Active Measurement (AM) value of AB
8MB File Transfer
time
Practical Passive Measurement (PPM) of L and Q: obtained from packet trace taken during 30 second Badabing measurement
Practical Passive Measurement (PPM) of AB: obtained from packet trace taken during Yaz measurement
Oracular Passive Measurements (OPM): AB, Q, and L obtained from packet trace taken during file transfer
Trace-based values of AB, Q, and L, both oracular and practical, are aggregates for all TCP traffic, not just the foreground TCP flow

Figure 4.2: Measurement traffic protocol, and oracular and practical path measurements used for SVR-based throughput prediction.

AB is defined as the spare capacity on the end-to-end path. YAZ estimates AB using an iterative method, so the time required to produce a single AB estimate can vary from a few seconds to tens of seconds.The passive measurement value of AB is computed by subtracting the observed traffic rate on the link from the realizable OC3 link bandwidth.

BADABING requires the sender and receiver to be time-synchronized. To accommodate our wide area experiments, the BADABING receiver was modified to reflect probes back to the sender, where they were timestamped and logged as on the original receiver. Thus, the sender clock was used for all probe timestamps. BADABING reports loss in terms of *loss episode frequency* and *loss episode duration*, defined in [115]. Although we refer to *loss frequency* and *loss duration* together as L or loss for expositional ease, these two metrics are two different features for SVR. Q is defined as the difference between the RTT of the current BADABING probe and the minimum probe RTT seen on the path. We set the BADABING probe probability parameter p to 0.3. Other BADABING parameters were set as described in Section 4.5. Passive values of Q and L are computed using the same definitions as above, except that they are applied to all TCP traffic on the bottleneck link instead of only BADABING probes.

As shown in Figure 4.2, the measurement protocol is the following:

1. Run BADABING for 30 seconds.
2. Run YAZ to obtain an estimate of the available bandwidth.
3. Transfer a file.

In the remainder of this chapter, we refer to the above series of steps as a single *experiment*, and to a number of consecutive experiments as a series. Experiments in the wide area omit the available bandwidth measurement. Individual experiments in a series are separated by a 30 second period. Series of experiments differ from each other in that either the background traffic is different between series, or the distribution of file sizes transferred is different, or the experiments are conducted over different physical paths. Each series of experiments is divided into two mutually exclusive training and test sets for the SVR. The SVR mechanism does not require that the sets of experiments that form the training and test set be consecutive or contiguous in time. In contrast, history-based prediction methods generally require consecutive historical information since they rely on standard timeseries-based forecasting models. In our evaluation we use contiguous portions for training and test sets, *i.e.* the beginning part of a series becomes the training set and the rest the test set. The number of experiments in training and test sets may be the same or different. Notions of separate training and test data sets are not required for history-based methods; rather, predictions are made over continuous notion of distant and recent history. In our evaluation of history-based methods, we use the final prediction of the training set as the starting point for the test set.

From each series of experiments, we gather three different sets of measurements. The first set, which we call *Oracular Passive Measurements (OPM)*, are AB, Q and L measurements during a file transfer that we obtain from packet traces. We refer to these measurements as *oracular* because they give us essentially perfect information about network conditions. In practice, this oracular information would not be available when making a prediction. We use this information to establish the best possible accuracy of our prediction mechanism. The second set, *Active Measurements (AM)*, are the measurements from our active measurement tools. Note that, unlike the *OPM,* the *AM* provide AB, Q and L values before the actual transfer. The third set, *Practical Passive Measurements (PPM)*, are trace-based measurements of AB, Q and L taken at the same time as *AM* are taken. Their purpose is to show how much better the prediction accuracy would be if the active measurements taken before a target transfer had perfect accuracy. It should be noted that all passive measurements are aggregates for conditions on the path rather than being specific to any single TCP flow.

We use the SVM^{light} package [63] to construct our SVR-based predictor. The input features for the SVR-based predictors are AB, Q, L and file size, and the target value is the TCP throughput. We scale or *standardize* each input feature by subtracting the feature mean and then dividing by the feature standard deviation The goal of standardization is to prevent features with a larger original scale from having a bigger impact on the solution than those with a smaller scale. We use a Radial Basis Function (RBF) kernel (Chapter 3, Section 3.5) because RBF kernels tend to have the best accuracy in practice when the relationship between input features and the target value are known to be non-linear. We set the SVM^{light} parameter C from Equation 3.34 to 3.162 and the RBF kernel parameter γ to 0.3162 based on ten-fold cross-validation of our laboratory data. We have found these values to result in high preduction accuracy for a variety of wide-area paths in addition to our laboratory environment.

For experiments in the wide area, we created a tool called PATHPERF. This tool, designed to run between a pair of end hosts, initiates TCP file transfers and path property measurements (using our modified version of BADABING), and produces throughput estimates using our SVR-based method. It can be configured to generate arbitrary file size transfers for both training and testing and initiates retraining when level shifts are identified as described in Section 4.5.

4.2.3 Evaluating Prediction Accuracy

We denote the actual throughput by *R* and the predicted throughput by $\hat{R}$. We use the metric *relative prediction error E* introduced in [54] to evaluate the accuracy of an individual throughput prediction. *Relative prediction error* is defined as

$$E = \frac{\hat{R} - R}{min(\hat{R}, R)}$$

In what follows, we use the distribution of the absolute value of *E* to compare different prediction methods.

4.3 Building a Robust Predictior

This section describes how we developed, calibrated, and evaluated our prediction mechanism through an extensive set of tests conducted in our lab test-bed.

4.3.1 Calibration and Evaluation in the High Traffic Scenario

The first step in developing our SVR-based throughput predictor is to find the combination of training features which lead to the most accurate predictions over a wide variety of path conditions. We trained predictors using features vector that contained different combination of our set of target path measurements (AB, Q, L) and the measured throughput. We then compare the accuracy of the different predictors.

We also compare the accuracy of SVR to the exponentially weighted moving average (EWMA) History-Based Predictor (HB) described in [54]:

$$\hat{R}_{i+1} = \alpha R_i + (1-\alpha)\hat{R}_i$$

with an α value of 0.3.

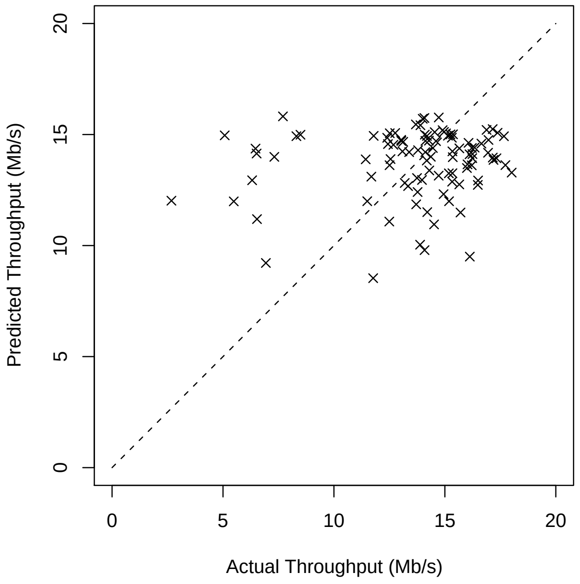
We do not report detailed results from tests with low utilization on the bottleneck link, *i.e.*, receive window bound flows, because in the low utilization scenarios there is very little variance in throughput of flows. Our SVR-based predictor generated 100% accurate forecasts for these tests. However, any reasonable prediction method can forecast throughput quite accurately in this scenario. For example, the formula-based predictor used in [54], which is based on [98], performs poorly in high utilization scenarios, but is accurate in low utilization scenarios.

For most of the results reported, we generated an average of 140 Mb/s of background traffic to create high utilization on our OC3 (155 Mb/s) bottleneck link. We used one set of 100 experiments for training and another set of 100 experiments for testing. In our initial experiments we transfer 8 MB files because this transfer size is large enough to make slow-start an insignificant factor in throughput for our receive window size of 128 KB. In later sections we consider smaller file sizes where slow-start has a bigger impact on overall throughput.

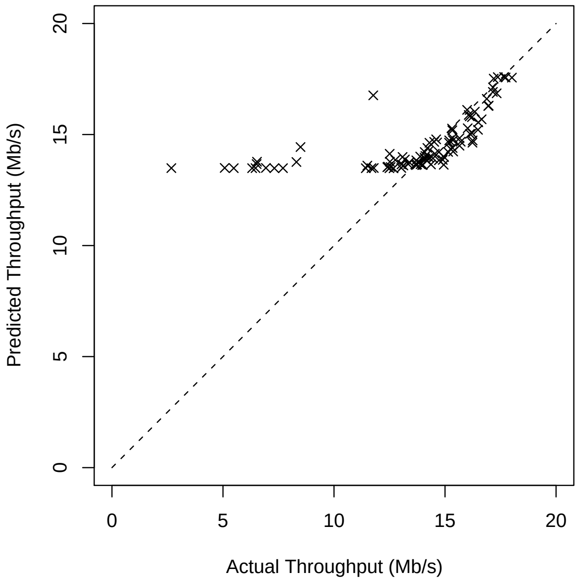
Figures 4.3a to 4.3l show scatter plots comparing the actual and predicted throughput using different prediction methods as discussed below. A point on the diagonal represents perfect prediction accuracy; the farther a point is from the diagonal, the greater the prediction error.

4.3.1.1 Using Path Measurements from an Oracle

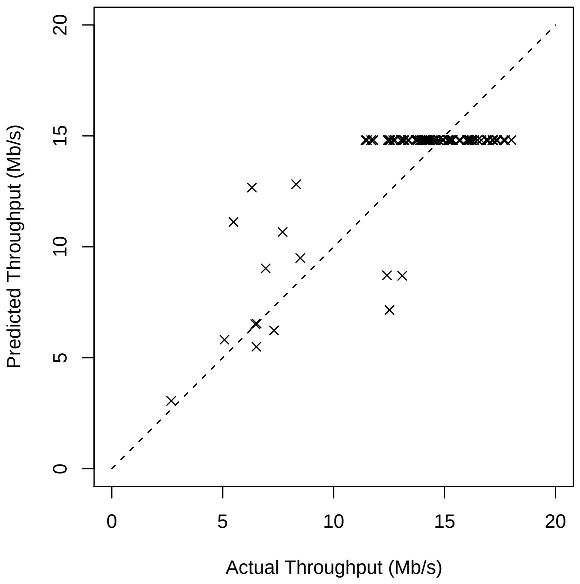
Figure 4.3a shows the prediction accuracy scatter plot for the HB method. Figures 4.3b to 4.3h show the prediction error with SVR using *Oracular Passive Measurements (OPM)* for different combinations of path measurements in the feature vector. For example, *SVR-OPM-Queue* means that only queuing delay measurements were used to train and test the predictor, while *SVR-OPM-AB-Queue* means that both available bandwidth and queuing delay measurements were used.



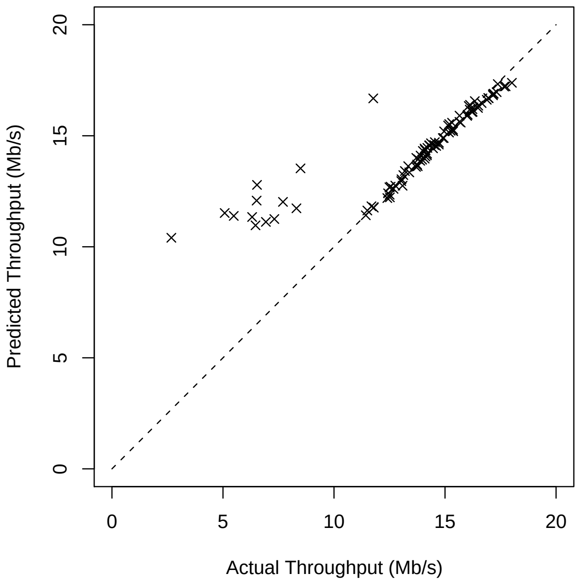
(a) HB



(b) SVR-OPM-AB

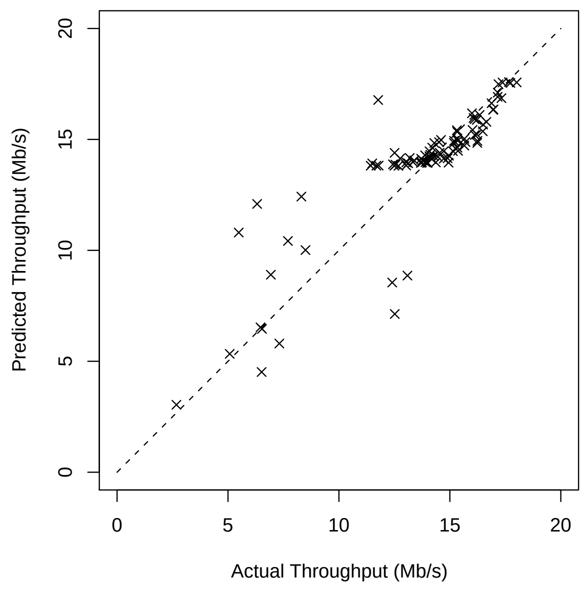


(c) SVR-OPM-Loss

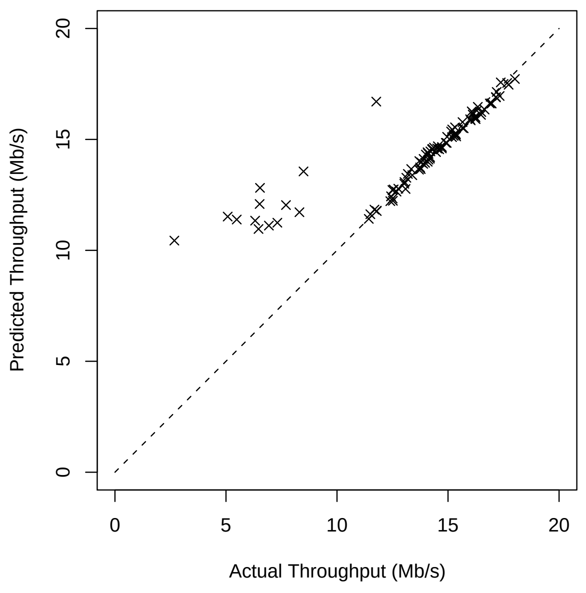


(d) SVR-OPM-Queue

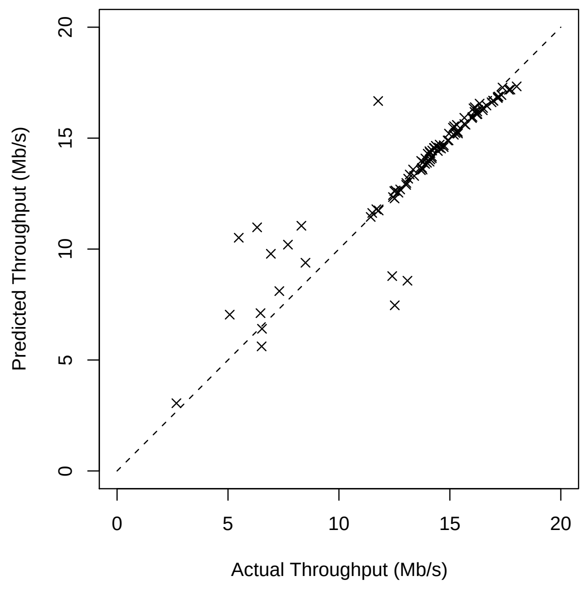
Figure 4.3: Comparison of prediction accuracy of HB, SVR-OPM, SVR-PPM and SVR-AM for the high background traffic scenario.



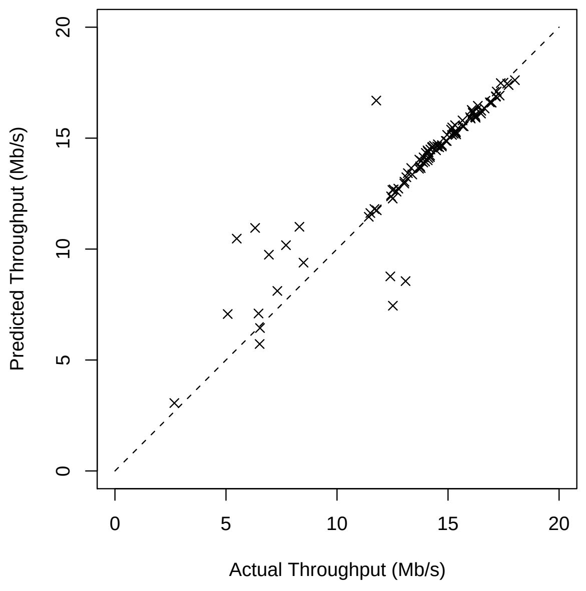
(e) SVR-OPM-AB-Loss



(f) SVR-OPM-AB-Queue

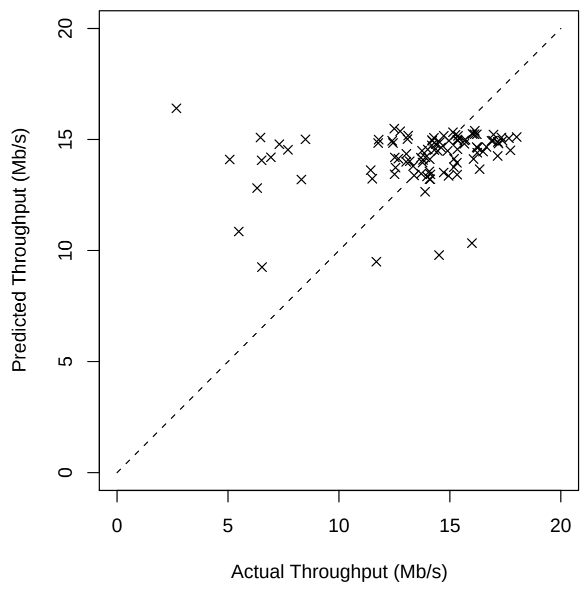


(g) SVR-OPM-Loss-Queue

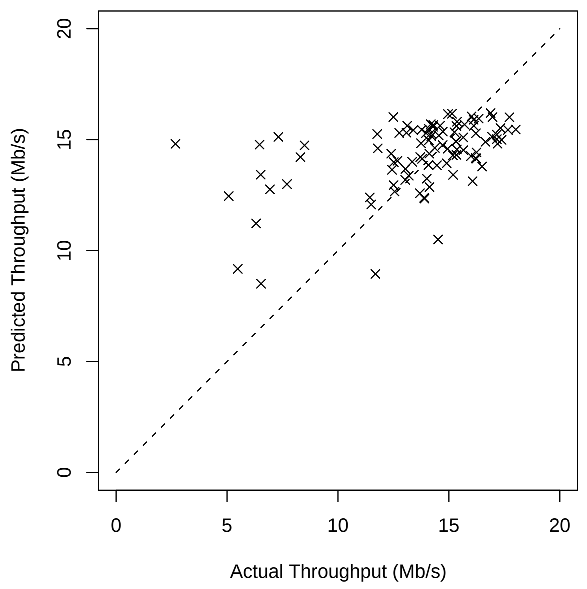


(h) SVR-OPM-AB-Loss-Queue

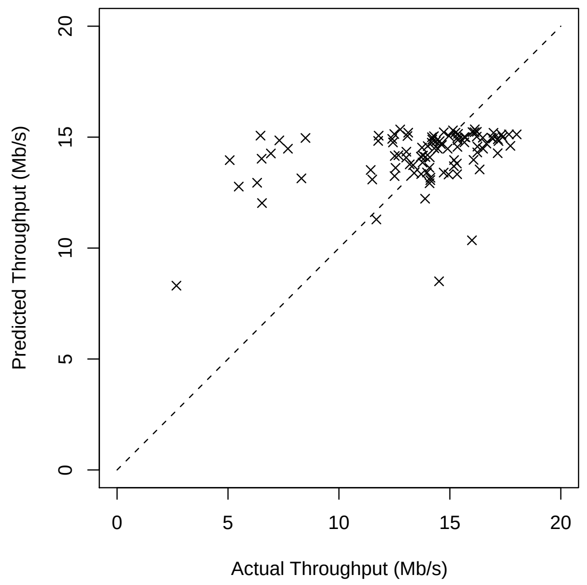
Figure 4.3: Comparison of prediction accuracy of HB, SVR-OPM, SVR-PPM and SVR-AM for the high background traffic scenario, continued.



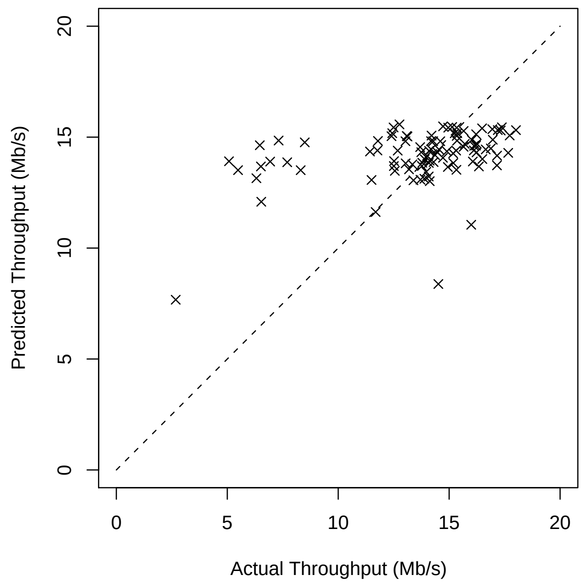
(i) SVR-PPM-Loss-Queue



(j) SVR-PPM-AB-Loss-Queue



(k) SVR-AM-Loss-Queue



(l) SVR-AM-AB-Loss-Queue

Figure 4.3: Comparison of prediction accuracy of HB, SVR-OPM, SVR-PPM and SVR-AM for the high background traffic scenario, continued.

Table 4.1 shows relative prediction errors for HB forecasting and for *SVM-OPM*-based predictions. Values in the table indicate the fraction of predictions for a given method within a given accuracy level. For example, the first two columns of the first row in Table 4.1 mean that 32% of HB predictions have relative prediction errors of 10% or smaller while 79% of *SVR-OPM-AB* predictions have relative prediction errors of 10% or smaller. We present scatter plots in addition to tabular data to provide insight into how different path properties contribute to throughput prediction in the SVR method.

From Figure 4.3a, we can see that the predictions for the HB predictor are rather diffusely scattered around the diagonal, and that predictions in low-throughput conditions tend to have large relative error. Figures 4.3b, 4.3c, and 4.3d show the behavior of the SVR predictor using a single measure of path properties in the feature vector—AB, L, and Q respectively. The AB and Q graphs have a similar overall trend: predictions are accurate (*i.e.*, points are close to the diagonal) for high actual throughput values, but far from the diagonal, almost in a horizontal line, for lower values of actual throughput. The L graph has the opposite trend: points are close to the diagonal for lower values of actual throughput, and form a horizontal line for higher values of actual throughput.

The explanation for these trends lies in the fact that file transfers with low actual throughput experience loss, while file transfers with high actual throughput do not experience any loss. When loss occurs, the values of AB and Q for the path are nearly constant. AB is almost zero, and Q is the maximum possible value (which depends on the amount of buffering available at the bottleneck link in the path). In this case, throughput depends on the value of L. Hence, L appears to be a good predictor when there is loss on the path, and AB and Q, being constants in this case, have no predictive power, resulting in horizontal lines, *i.e.*, a single value of predicted throughput. On the other hand, when there is no loss, L is a constant with value zero, so L has no predictive power, while AB and Q are able to predict throughput quite accurately.

Figures 4.3e to 4.3h show improvements on the prediction accuracy obtained by using more than one path property in the SVR feature vector. We can see that when L is combined with either AB or Q, the horizontal lines on the graphs are replaced by points much closer to the diagonal, so combining L with AB or Q allows the SVR method to predict accurately in both lossy and lossless network conditions.

Measurements of AB and Q appear to serve the same function, namely, helping to predict throughput in lossless conditions. This observation begs the question: do we really need both AB and Q, or can we use just one of the two and still achieve the same prediction accuracy? To answer this question, we compared *AB-Loss* and *Loss-Queue* predictions with each other and with *AB-Loss-Queue* predictions (*i.e.*, Figures 4.3e, 4.3g, and 4.3h). The general trend in all three cases, as seen from the scatter plots, is the same: the horizontal line of points is reduced or eliminated, suggesting that prediction from non-constant-value measurements is occurring for both lossy and lossless network conditions. If we compare the *AB-Loss* and *Loss-Queue* graphs more closely, we observe two things. First, in the lossless prediction case, the points are closer to the diagonal in the *Loss-Queue* case than in the *AB-Loss* case. Second, in the *Loss-Queue* case, the transition in the prediction from the lossless to the lossy case is smooth, *i.e.*, there is no horizontal line of points, while in the *AB-Loss* case there is still a horizontal line of points in the actual throughput range of 11–14 Mbps. This suggests that Q is a more accurate predictor than AB in the lossless case. The relative prediction error data of Table 4.1 supports this: SVR with a feature vector containing *Loss-Queue* information predicts throughput within 10% of actual for 87% of transfers, while a feature vector containing *AB-Loss* measurements predicts with the same accuracy level for 78% of transfers. Finally, there is no difference in accuracy (either qualitatively or quantitatively) between *Loss-Queue* and *AB-Loss-Queue*.

The above discussion suggests that AB measurements are not required for highly accurate throughput prediction, and that given our framework, a combination of L and Q is sufficient. This observation is not only surprising, but rather good news. Prior work has shown that accurate measurements of AB require at least moderate amounts of probe traffic [116, 112], and formula-based TCP throughput estimation takes as a given that AB measurements are necessary for accurate throughput prediction [54]. In contrast, measurements of L and Q can be very lightweight probe processes [115]. We discuss measurement overhead further in Section 4.5.

4.3.1.2 Using Practical Passive and Active Path Measurements

So far, we have considered the prediction accuracy of SVR based on only *Oracular Passive Measurements (OPM)*. This gives us the best-case accuracy achievable with SVR, and also provides insight into how SVR uses different path properties for prediction. Table 4.1 shows that HB predicts 32% of transfers within 10% of actual while *SVR-OPM-Loss-Queue* predicts 87%, an almost 3-fold improvement. In practice, however, perfect measurements of path properties are not available, so in what follows we assess SVR using measurements that can be obtained in practice in the wide area.

We compare HB accuracy with *SVR-PPM* accuracy and *SVR-AM* accuracy. We focus our discussion on *Loss-Queue* and *AB-Loss-Queue* results for SVR. We choose these two combinations because we expect *AB-Loss-Queue* to have the highest accuracy as it has the most information about path properties, and *Loss-Queue* because it is very lightweight and has accuracy equal to *AB-Loss-Queue* for *SVR-OPM*.

Figures 4.3i and 4.3j show predicted and actual throughput for *SVR-PPM* and Figures 4.3k and 4.3l show predicted and actual throughput for *SVR-AM*. Table 4.2 presents relative prediction error data for HB, *SVR-PPM* and *SVR-AM*. We wish to examine three issues: first, whether our finding that *Loss-Queue* has the same prediction accuracy as *AB-Loss-Queue* from the *SVR-OPM* case holds for the *SVR-PPM* and *SVR-AM* case; second, whether *SVR-PPM* and *SVR-AM* have the same accuracy; and third, how *SVR-AM* accuracy compares with HB prediction accuracy.

All the scatter plots from Figures 4.3i to 4.3l are qualitatively very similar. This is encouraging because it suggests two things. First, *Loss-Queue* has prediction accuracy similar to *AB-Loss-Queue*, *i.e.*, the observation from *SVR-OPM* still holds, so we can achieve good prediction accuracy without having to measure AB. The relative prediction error data in Table 4.2 supports this graphical observation: *AB-Loss-Queue* has only slightly better accuracy than *Loss-Queue* for both *SVR-PPM* and *SVR-AM*. Second, *SVR-AM* has accuracy similar to *SVR-PPM*, *i.e.*, using

Table 4.1: Relative accuracy of history-based (HB) throughput prediction and SVR-based predictors using different types of oracular passive path measurements (*SVR-OPM*) in the feature vector. Table values indicate the fraction of predictions within a given accuracy level.

Relative Error	***HB***	***AB***	***L***	***Q***	***AB-L***	***AB-Q***	***L-Q***	***AB-L-Q***
10%	0.32	0.79	0.54	0.87	0.78	0.87	0.86	0.86
20%	0.67	0.87	0.86	0.87	0.87	0.87	0.90	0.90
30%	0.80	0.87	0.92	0.87	0.91	0.87	0.90	0.90
40%	0.87	0.87	0.94	0.87	0.92	0.87	0.93	0.93
50%	0.88	0.88	0.95	0.89	0.97	0.89	0.96	0.96
60%	0.88	0.88	0.97	0.92	0.97	0.92	0.97	0.97
70%	0.89	0.89	0.97	0.94	0.97	0.94	0.98	0.98
80%	0.92	0.91	0.98	0.95	0.98	0.95	0.99	0.99
90%	0.92	0.92	0.98	0.96	0.98	0.96	0.99	0.99

Table 4.2: Relative accuracy of history-based (HB) throughput prediction and SVR-based predictors using trace-based passive path measurements (*PPM*) or active path measurements (*AM*). Table values indicate the fraction of predictions within a given accuracy level.

Relative Error	***HB***	***PPM***		***AM***	
		L-Q	***AB-L-Q***	***L-Q***	***AB-L-Q***
10%	0.32	0.49	0.53	0.49	0.51
20%	0.67	0.77	0.81	0.78	0.76
30%	0.80	0.86	0.86	0.86	0.86
40%	0.87	0.86	0.89	0.86	0.86
50%	0.88	0.88	0.89	0.86	0.87
60%	0.88	0.90	0.89	0.88	0.87
70%	0.89	0.90	0.91	0.88	0.88
80%	0.92	0.91	0.94	0.90	0.90
90%	0.92	0.92	0.95	0.92	0.92

Table 4.3: Relative accuracy of history-based (*HB*) and SVR-based throughput predictors using oracular path measurements (*OPM*) and active measurements (*AM*) when predictors are subjected to shifts in average background traffic volume.

Relative Error	***HB***	***OPM-L-Q TrainLowTestHigh***	***OPM-Loss-Queue***	***AM-Loss-Queue***
10%	0.29	0.00	0.62	0.52
20%	0.40	0.00	0.72	0.61
30%	0.52	0.00	0.81	0.69
40%	0.55	0.00	0.84	0.73
50%	0.62	0.00	0.88	0.77
60%	0.64	0.00	0.90	0.78
70%	0.66	0.00	0.91	0.80
80%	0.71	0.00	0.92	0.83
90%	0.74	0.00	0.92	0.84

Table 4.4: Comparison of relative accuracy of SVR-based throughput prediction using active measurements (*AM*) and different numbers of training samples. 40 training samples are used in *AM-Loss-Queue*, 5 samples for *AM-L-Q-5*, and 10 samples for *AM-L-Q-10*. Background traffic consists of shifts in average traffic volume between 120 Mb/s and 160 Mb/s.

Relative Error	***AM-Loss-Queue***	***AM-L-Q-5***	***AM-L-Q-10***
10%	0.52	0.44	0.45
20%	0.61	0.49	0.53
30%	0.69	0.51	0.55
40%	0.73	0.55	0.59
50%	0.77	0.58	0.62
60%	0.78	0.60	0.65
70%	0.80	0.62	0.67
80%	0.83	0.65	0.73
90%	0.84	0.67	0.73

Table 4.5: Relative accuracy of *SVR*-based predictor using oracular passive measurements (*OPM*) and training sets consisting of 1, 2, 3, 6, or 8 distinct file sizes.

Relative Error	**No. of distinct file sizes in training**				
	1	**2**	**3**	**6**	**8**
10%	0.06	0.24	0.49	0.35	0.34
20%	0.16	0.40	0.57	0.48	0.51
30%	0.18	0.52	0.64	0.54	0.54
40%	0.19	0.61	0.66	0.59	0.61
50%	0.22	0.64	0.67	0.65	0.66
60%	0.24	0.67	0.68	0.66	0.67
70%	0.24	0.69	0.68	0.67	0.67
80%	0.29	0.71	0.69	0.68	0.68
90%	0.30	0.72	0.70	0.68	0.68

Table 4.6: Relative accuracy of *SVR*-based predictor using active measurements (*AM*) and training sets consisting of 1, 2, 3, 6, or 8 distinct file sizes.

Relative Error	**No. of distinct file sizes in training**				
	1	**2**	**3**	**6**	**8**
10%	0.10	0.29	0.40	0.29	0.29
20%	0.15	0.41	0.51	0.47	0.47
30%	0.16	0.53	0.59	0.52	0.55
40%	0.19	0.58	0.64	0.57	0.61
50%	0.23	0.64	0.65	0.62	0.64
60%	0.23	0.66	0.66	0.64	0.65
70%	0.26	0.70	0.67	0.64	0.66
80%	0.28	0.70	0.68	0.64	0.67
90%	0.31	0.71	0.69	0.65	0.68

active measurement tools to estimate path properties yields predictions almost as accurate as having ground-truth measurements. Similar accuracy between *SVR-PPM* predictions and *SVR-AM* predictions is important because in real wide-area Internet paths instrumentation is not available for providing accurate passive measurements.

Finally, we compare HB with *SVR-AM*. Although Figures 4.3a, 4.3k and 4.3l are qualitatively similar, *SVR-AM* has a tighter cluster of points around the diagonal for high actual throughput than HB. Thus, *SVR-AM* appears to have higher accuracy than HB. As Table 4.2 shows, *SVR-AM-Loss-Queue* predicts throughput within 10% of actual accuracy 49% of the time, while HB does so only 32% of the time. Hence, for high traffic scenarios, *SVR-AM-Loss-Queue*, the practically deployable lightweight version of the SVR-based prediction mechanism, significantly outperforms HB prediction.

The above observations raise the question of why SVR performs so much better than HB at high accuracy levels. HB is a weighted average that works on the assumption that transfers close to each other in time will have similar throughputs. HB loses accuracy when there are rapid and large changes in throughput, because in that case there is a large difference between the weighted average and individual throughputs. SVR predicts throughput by constructing a function mapping path properties to throughput values. Due to the use of kernels (Chapter 3, Section 3.5), this function is able to capture complex non-linear relationships between the path properties and throughput. As long as the mapping between path properties and resulting throughput is consistent, SVR can predict throughput accurately, regardless of the variation between the throughput of neighboring transfers. SVR accuracy breaks down only when the mapping between path properties and throughput becomes inconsistent, *e.g.*, when path conditions are so volatile that they change between the active path measurement and the file transfer, or when an aggregate path measure does not hold for the tranfer, such as when there is loss on the bottleneck link but the file transfer experiences more or less loss than the aggregate traffic. Thus, SVR outperforms HB because *(a)* it has access to additional information, in the form of path properties such as AB, L, and Q, while HB has access only to past throughputs, and *(b)* it constructs a much more sophisticated functional mapping of path properties to throughput compared to the simple time-series analysis that HB uses. The fact that SVR's supremacy over HB is limited to high accuracy levels, and HB catches up with SVR for lower accuracy levels (predicted throughput within 40%-90% of actual, Table 4.2), is a feature of the distribution of throughputs in our test set. We can see from the graphs in Figure 4.3a- 4.3k that the average throughput of the test set is between 10 and 15 Mb/s. We expect the HB prediction to be around the average throughput. Figure 4.3a shows that this is indeed the case: all HB predictions are between 10 and 15 Mb/s. All but one transfers have actual throughput between 5 Mb/s and 18 Mb/s, so the relative error is 100% or less ($\frac{10-5}{5} * 100 = 100$) for low throughputs (all predictions within a factor of two of actual) and 80% or less ($\frac{18-10}{10} * 100 = 80$) for high throughputs. If the range of actual throughputs had been wider but the average had been the same, HB would have been less able to match SVR accuracy for relative error values of 40%-90%.

A remaining question is why Q is a better predictor of throughput than AB. When we compare AB and Q prediction behavior in Figures 4.3b and 4.3d, we see that as throughput decreases, *i.e.*, as the network moves from lossless to lossy conditions, AB loses its predictive power sooner than Q does, around 15 Mb/s instead of 11 Mb/s for Q. As soon as the link experiences loss, AB is zero. While Q eventually reaches a maximum value under sufficiently high congestion, under light loss it retains its resolution because Q will be different depending on the percentage of packets that experience the maximum possible queuing delay and those which experience less than the maximum. Hence Q is a better predictor of throughput than AB most likely because it can be measured with better resolution for a wider range of network conditions.

4.3.1.3 The Nature of Prediction Error

Lu *et al.* [81] observed in their study that throughput prediction errors were approximately normal in distribution. As the authors noted, normality would justify standard computations of confidence intervals. We examined the distribution of errors in our experiments and also found evidence suggestive of normality.

Figure 4.4 shows two normal quantile-quantile (Q-Q) plots for *SVR-OPM-Loss-Queue* (Figure 4.4a) and *SVR-AM-Loss-Queue* (Figure 4.4b). Samples that are consistent with a normal distribution form approximately a straight line in the graph, particularly toward the center. In Figures 4.4a and 4.4b, we see that the throughput prediction samples in each case form approximately straight lines. These observations are consistent with normality in the distribution of prediction errors. Error distributions from predictors based on other combinations of measurements were also consistent with the normal distribution. We further discuss the issues of retraining and of detecting estimation problems in Section 4.5.

4.3.2 Evaluation of Prediction Accuracy with Level Shifts in Background Traffic

In the previous section, we considered SVR and HB prediction accuracy under variable but stationary background traffic. In this section, we consider the case where there is a shift in average load of background traffic.

We configure our traffic generation process to cycle between 120 Mb/s and 160 Mb/s average offered loads. With 120 Mb/s traffic, there is virtually no loss along the path and the throughput is bound by the receive window size. With 160 Mb/s traffic, the OC3 bottleneck is saturated and endemic loss ensues. The procedure is to run 120 Mb/s background traffic for 75 minutes, followed by 160 Mb/s of traffic for 2 hours 15 min, repeating this cycle several times. We follow the measurement protocol in Section 4.2.2 for collecting AB, L, Q, and throughput measurements. We first train our SVR predictor in the 120 Mb/s environment, and test it in the 160 Mb/s environment. The column labeled *OPM-L-Q TrainLowTestHigh* in Table 4.3 shows the results. Clearly, the prediction accuracy is very poor: all predictions are off by a factor of two or more. Next, we train the predictor on one whole cycle of 120 Mb/s and 160 Mb/s, and test it on a series of cycles. The last two columns of Table 4.3, *OPM-Loss-Queue* and *AM-Loss-Queue*, and Figures 4.5a and 4.5b show the results from these experiments. The x-axis in Figures 4.5a and 4.5b represents time in terms of the number of completed file transfers, and the y-axis is throughput. In this case, both *SVR-OPM* and

SVR-AM predict throughput with high accuracy. *SVR-OPM* predicts throughput within 10% of actual 62% of the time, and *SVR-AM* 52% of the time.

For comparison, the first column of Table 4.3 and Figure 4.5c present results for the history-based predictor. HB accuracy is significantly lower than SVR accuracy, only about half as much as *SVR-OPM*. Figure 4.5c explains this disparity. Every time there is a level-shift in the background traffic, the HB predictor takes time to re-adapt. In contrast, there is no adaptation time required for the SVR predictor if it has already been trained on the range of expected traffic conditions.

One issue regarding shifts in average traffic volume is how many samples from a new, previously unseen traffic level are needed to adequately train the predictor. To examine this issue, we train our predictor on two new kinds of average traffic loads: a step constituting 75 minutes of 120 Mb/s traffic followed by 10 minutes of 160 Mb/s traffic resulting in 5 experiments at the higher traffic level, and a step constituting of 75 minutes of 120 Mb/s traffic followed by 20 minutes of 160 Mb/s traffic resulting in 10 experiments at the higher traffic level. We call these conditions *SVR-AM-L-Q-5* and *SVR-AM-L-Q-10* respectively. The prediction accuracy of these schemes is presented in Table 4.4. As can be seen in the table, these training schemes yield most of the accuracy of *SVR-AM-L-Q*, which trains at the 160 Mb/s level for approximately 40 experiments. These results demonstrate that fairly small training sets from new traffic levels are needed for accurate prediction. A comparison of time series plots from these experiments provides further insight into how prediction accuracy improves with larger training sets. Both *SVR-AM-L-Q-5* and *SVR-AM-L-Q-10* have periods where the predicted throughput is virtually constant even though the actual throughput is not. Such periods are shorter for *SVR-AM-L-Q-10* compared to *SVR-AM-L-Q-5*. *SVR-AM-L-Q* (Figure 4.5b) has no such periods. The reason for this effect is that predictors incorporating fewer samples may not include a broad enough range of possible network conditions. However, the SVR prediction mechanism can still yield high accuracy using a small set of training samples if the samples are representative of the range of network conditions along a path.

4.3.3 Evaluation of Prediction Accuracy for Different File Sizes

Thus far, we have predicted throughput only for bulk transfers of 8 MB. However, our goal is accurate prediction for a range of file sizes, not just bulk transfers, so in this section we evaluate prediction accuracy for a broad range of file sizes, something not considered in prior HB prediction studies.

We conducted experiments using background traffic at an average offered load of 135 Mb/s. We used 9 different training sets, each of size 100, consisting of between 1 and 9 unique file sizes. The file sizes for the training sets were between 32 KB (2^{15}bytes) and 8 MB (2^{23}bytes). The first training set consisted of 100 8 MB files, the second of 50 32 KB files and 50 8 MB files, the third of 33 each of 32 KB (2^{15}bytes), 512 KB (2^{19}bytes) and 8 MB (2^{23}bytes) files, the fourth of 25 each of 2^{15}, $2^{17.66}$,$2^{20.33}$, and 2^{23} byte files, and so on, dividing the range of the exponent, 15 to 23, into more numerous but smaller intervals for each subsequent training set.

Test sets for our experiments consist of 100 file sizes between 2 KB (2^{11}bytes) and 8 MB (2^{23}bytes). To generate file sizes for the test set, uniformly distributed random numbers r between 11 and 23 were generated. The file size was set to 2^r bytes, rounded to the closest integer. This log-based distribution of file sizes contains a much greater number of smaller files than a uniform linear distribution of file sizes from 2 KB to 8 MB would. The bias towards smaller file sizes is important for proper evaluation of predictor performance for small files because it generates a more even distribution between files that spend all their transfer time in slow-start, those which spend some time in slow-start and some in congestion avoidance, and those which spend a negligible amount of time in slow-start. If the uniform linear distribution had been used, most of the file sizes generated would have spent a negligible amount of time in slow-start. We also use a wider range of small files in the test set compared to the training set – the test set starts with 2 KB files while the training set starts with 32 KB – because we want to see how well our predictor extrapolates for unseen ranges of small file sizes.

Tables 4.5 and 4.6 present prediction accuracy for training sets consisting of 1, 2, 3, 6, or 8 distinct file sizes for *SVR-OPM* and *SVR-AM*. Graphs in Figure 4.6 present *SVR-AM* results for one, two, and three file sizes in the training set. We do not include graphs for remaining training sets because they are similar to those for two and three file sizes in the training set. The first observation is that for the single file size in training, the prediction error is very high. This inaccuracy is expected because the predictor has been given no information about the relationship between size and throughput for small files. The second observation is that for more than one file size, prediction becomes dramatically more accurate, *i.e.*, the predictor is able to successfully extrapolate from a handful of sizes in training to a large number of sizes in testing. The third observation is that relative error is low for large file sizes (corresponding to high actual throughput) while it is higher for small files (low actual throughput), or, in other words, predicting throughput accurately for small files is more difficult than for large files. For small files, throughput increases rapidly with file size during TCP's slow-start phase due to the doubling of the window every RTT. Also, packet loss during the transfer of a small file has a larger relative impact on throughput. Due to the greater variability of throughput for small files, predicting throughput accurately is more difficult. The fourth observation is that for small file sizes (*i.e.*, small actual throughput), the error is always that of over-prediction. The smallest file in the training set is 32 KB while the smallest file in the test set is 2 KB. This difference is the cause of over-prediction errors: given the high variability of throughput for small file sizes due to slow-start effects and losses having a greater relative impact on throughput, without a broader training set, the SVR mechanism is unable to extrapolate accurately for unseen ranges of small file sizes.

A final observation is that prediction accuracy reaches a maximum at three file sizes in the training set, and there is no clear trend for four to nine file sizes in the training set. The number of transfers in our training set is always constant at 100, so for the single training size, there are 100 8 MB transfers, for the two training sizes, there are 50 32 KB transfers and 50 8 MB transfers, and so on. We believe that accuracy is maximum at three training sizes in our experiments because there is a trade-off between capturing a diversity of file sizes and the number of samples for a single file size. We would not see maximum accuracy occurring at three file sizes and would instead see an increase in accuracy with increasing number of file sizes in the training set if we kept the number of samples of a single file size constant at 100 in the training set and allowed the size of the training set to increase from 100 to 200, 300, etc., as we increase the number of file sizes in the training set.

4.4 Wide-Area Results

To further evaluate our SVR-based TCP throughput prediction method we created a prototype tool called PATHPERF that can generate measurements and make forecasts on wide area paths. We used PATHPERF to conduct experiments over a set of paths in the RON testbed [13]. This section presents the results of our wide area experiments.

4.4.1 The RON Experimental Environment

The RON wide area experiments were conducted in January 2007 over 18 paths between 7 different node locations. Two nodes were in Europe (in Amsterdam and London), and the remainder were located at universities in the continental United States (Cornell, Maryland, New Mexico, NYU, and Utah). Of the 18 paths, two are trans-European, 9 are trans-Atlantic, and 7 are trans-continental-US. The RON testbed has a significantly larger number of available nodes and paths, but two considerations limited the number of nodes and paths that we could use. The first consideration was that the nodes should have little or no other CPU or network load while our experiments were running: this is required for BADABING to measure loss accurately. The second issue was that we could not use any nodes running FreeBSD 5.x because the TCP throughput history caching feature in FreeBSD 5.x, controlled by the variable net.inet.tcp.inflight.enable, is on by default, and interfered with our experiments. Consequently, we were restricted to using nodes running FreeBSD 4.7, which does not have the throughput history caching feature.

We use the following convention for path names: A-B means that A is the TCP sender and B is the receiver, *i.e.*, the TCP data flow direction is from A to B, and TCP ack flow is in the reverse direction. A-B and B-A are considered two different paths, because routing and background traffic level asymmetry between the forward and reverse directions can lead to major differences in throughput.

The wide area measurement protocol was the following:

1. Run BADABING for 30 seconds,
2. Transfer a 2 MB file,
3. Sleep 30 seconds, and
4. Repeat the above 200 times.

In Section 4.3.1.1, we showed that available bandwidth measurements are not required for accurate TCP throughput prediction, so we omit running YAZ in the wide area. Thus, the wide area prediction mechanism is the *SVR-AM-Loss-Queue* protocol from Section 4.3.1.2. We reduced the size of the file transfers from 8MB in the lab experiments to 2 MB in the wide area experiments because the typical throughput in the wide area was an order of magnitude lower compared to the typical throughput in the lab (1-2 Mbps versus 10-15 Mbps).

The measurements were carried out at different times of the day for different paths depending upon when the nodes were available, *i.e.*, when the nodes had little or no other CPU and network activity in progress. Depending again on node availability, we ran a single set of experiments on some paths, and multiple sets on others. We can only conduct active measurements in the wide area because the infrastructure required for passive measurements is not present.

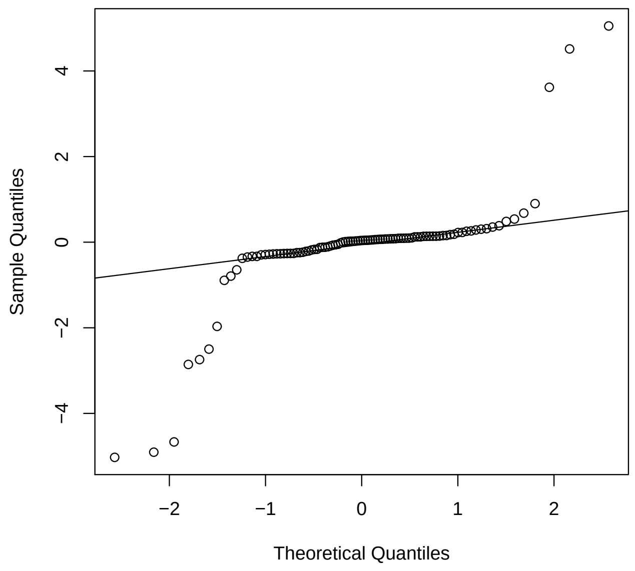
Table 4.7 lists the minimum round trip times observed on the wide area paths over all BADABING measurements taken on each path. The shortest paths have a minimum RTT of 8ms and the longest a minimum RTT of 145 ms. In Table 4.7 we ignore path directions, *e.g.*, we list Amsterdam-London and London-Amsterdam as a single entry, because minimum RTT is the same in both directions.

4.4.2 Wide Area Results

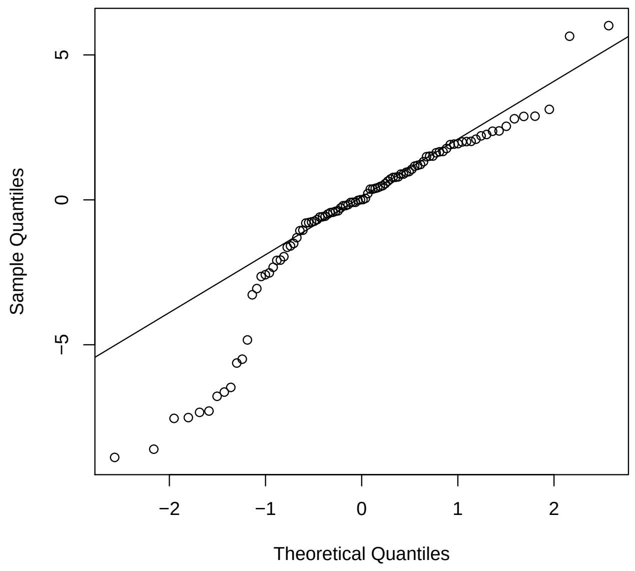
Figure 4.7 compares the accuracy of the SVR and HB throughput predictions for the 18 wide area paths. The results in Figure 4.7 are obtained by dividing the 200 measurements gathered for each path into two consecutive sets of 100, and using the first 100 as the training set and the second 100 as the test set. Some paths feature more than once because node availability allowed us to repeat experiments on those paths.

Table 4.7: Minimum RTTs for the 18 RON Paths used for wide-area experiments

Paths (Abbreviations)	**Minimum RTT (ms)**
Amsterdam-London (A-L, L-A)	8
Cornell-Amsterdam (C-A)	92
Cornell-London (C-L)	83
Cornell-NYU (C-NY)	8
Cornell-Utah (C-U, U-C)	71
London-Maryland (L-M)	81
London-NYU (L-NY, NY-L)	72
London-Utah (L-U, U-L)	145
New Mexico-Maryland (NM-M)	106
New Mexico-Cornell (NM-C)	87
NYU-Amsterdam (NY-A)	80
NYU-London (NY-L)	62
NYU-Utah (NY-U, U-NY)	67

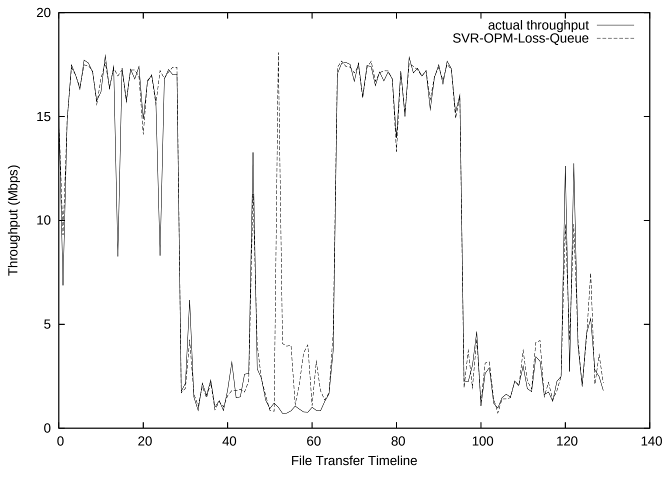


(a) Q-Q Plot *SVR-OPM-Loss-Queue*.

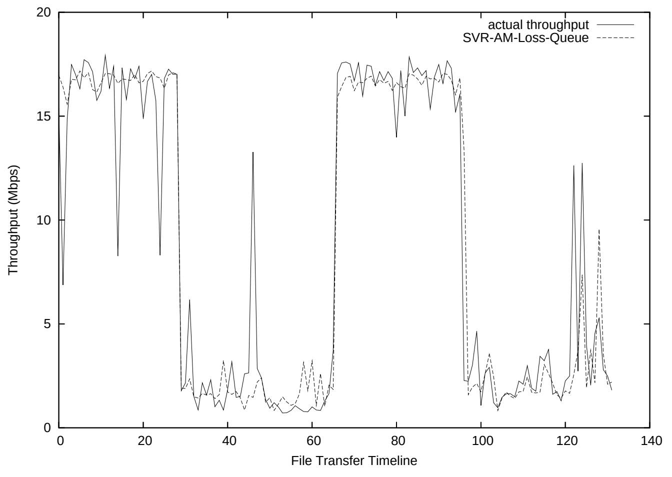


(b) Q-Q Plot for *SVR-AM-Loss-Queue*.

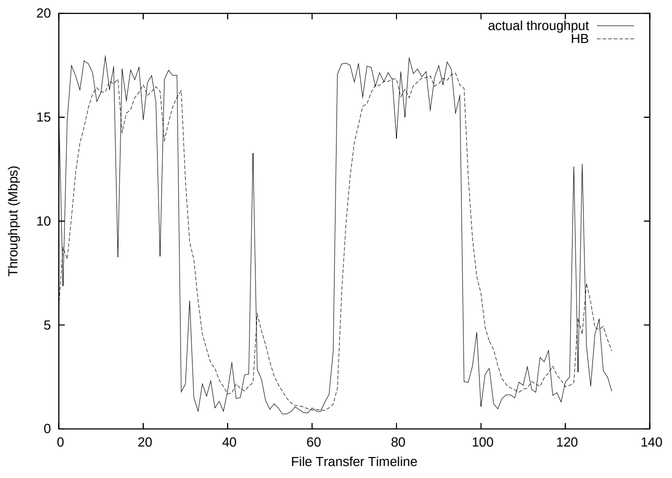
Figure 4.4: Normal Q-Q Plots for prediction errors with oracular (a) and active (b) measurements of *Loss-Queue*.



(a) Level Shift: *SVR-OPM-Loss-Queue*

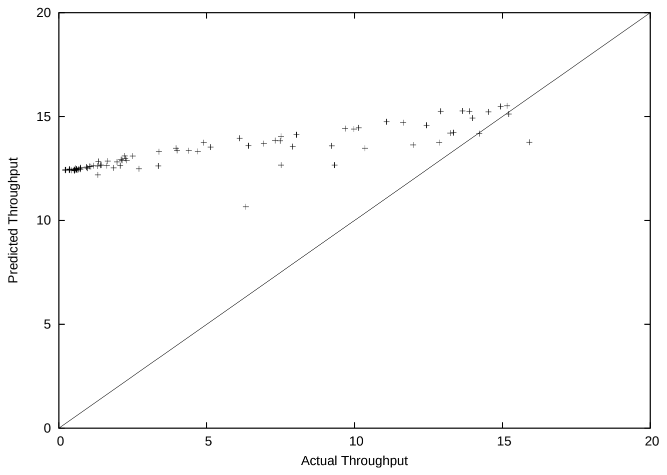


(b) Level Shift: *SVR-AM-Loss-Queue*

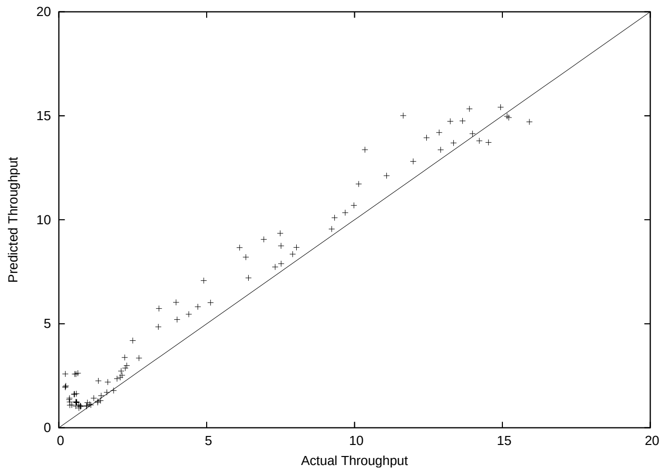


(c) Level Shift: HB

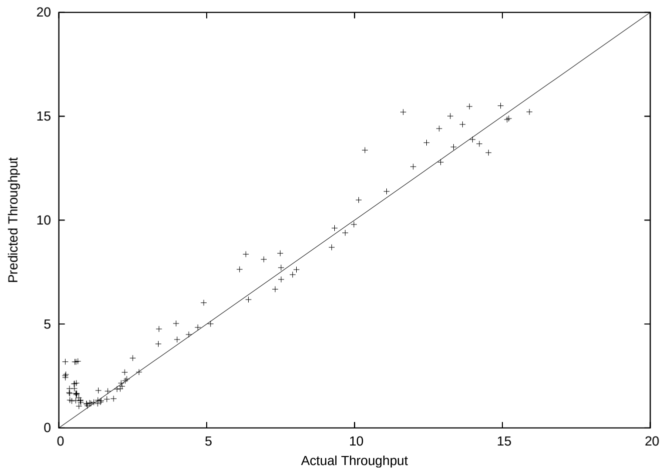
Figure 4.5: Prediction accuracy in fluctuating background traffic



(a) *SVR-AM* with file size of 8 MB in training set.



(b) *SVR-AM* with file sizes of 32 KB and 8 MB in training set.



(c) *SVR-AM* with file sizes of 32 KB, 512 KB, and 8 MB in training s et.

Figure 4.6: Scatter plots for the *SVR*-based predictor using 1, 2, or 3 distinct file sizes in the training set. All results shown use active measurements to train the predictor. Testing is done using a range of 100 file sizes from 2 KB to 8 MB.

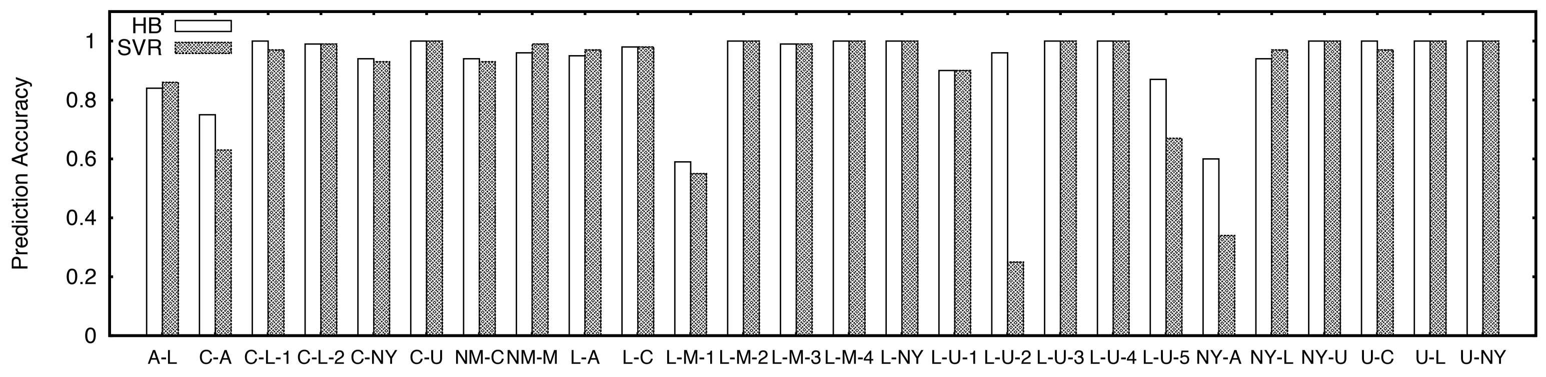


Figure 4.7: Wide Area Results for HB and SVR predictions: the HB and SVR columns present the fraction of predictions with relative error of 10% or less. Training and test sets consisted of 100 samples each. 2MB of data was transferred. Labels use node initials; see Section 4.7 for complete node names.

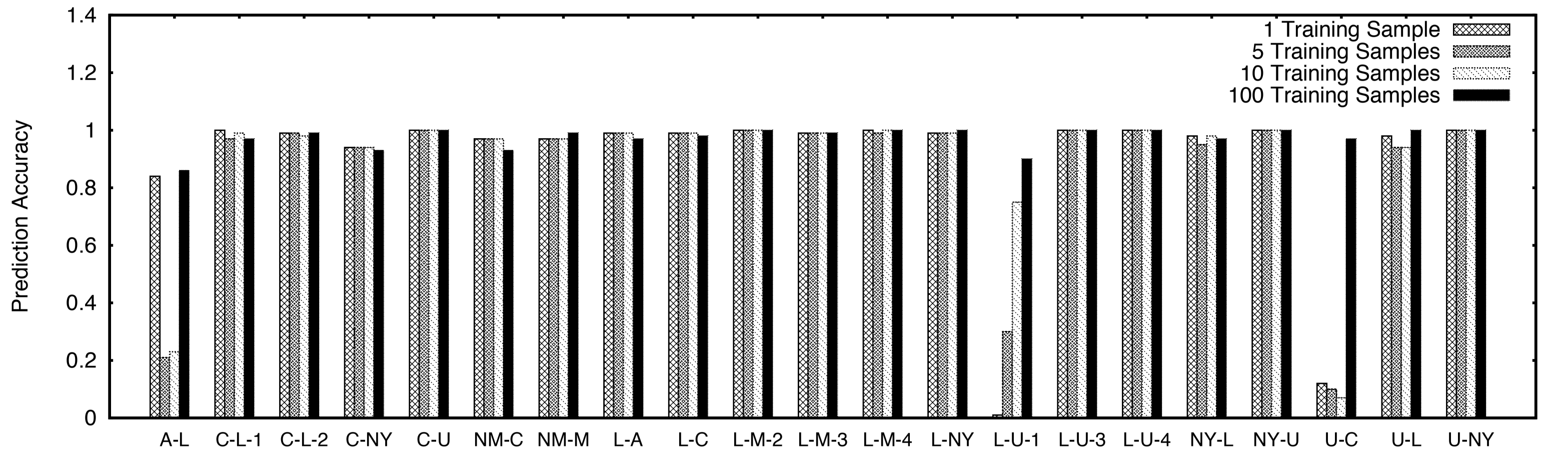


Figure 4.8: Effect of training set size on SVR prediction accuracy in the wide area. For paths with high prediction accuracy in Figure 4.7, this table shows how large a training set size was needed to achieve high accuracy. Labels use node initials; see Section 4.7 for complete node names.

We observe two major trends from Figure 4.7. First, for the majority of experiments, prediction accuracy is very high for both HB and SVR: most paths have greater than 85% of predictions within 10% of actual throughput. Second, for 5 out of the 26 experiments – Cornell-Amsterdam, London-Maryland-1, London-Utah-2, London-Utah-5 and NYU-Amsterdam – the SVR prediction accuracy is quite low, as low as 25% for London-Utah-2. The reasons for this poor accuracy will be analyzed in detail in Section 4.4.3.

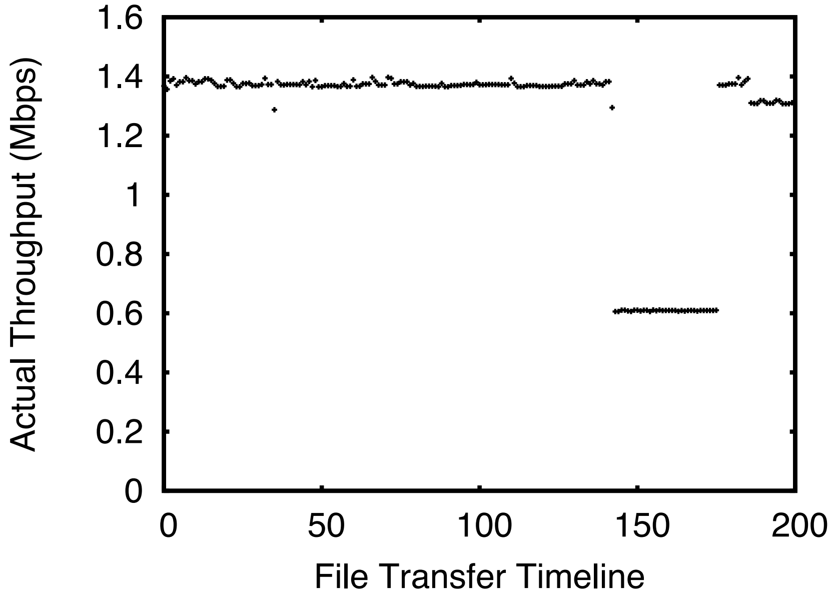
Next, we take a more detailed approach to assessing prediction accuracy. We divide the 200 samples into sets of k and $200-k$ for values of k from 1 to 199. The first k samples are used for training, and the remaining $200-k$ samples are used for testing. This allows us to understand the trade-off between training set size and prediction accuracy, which is important for a practical online prediction system where it is desirable to start generating predictions as soon as possible. This analysis also allows us to identify the points in a trace where an event that has an impact on prediction accuracy, such as a level shift, occurs, and whether retraining helps maintain prediction accuracy in the face of changing network conditions. We present this data for SVR only, as for HB, there is no division of data into training and test sets because retraining occurs after every measurement.

Figure 4.8 presents the SVR prediction accuracy for $k = 1$, 5, and 10 for those experiments in Figure 4.7 that had high prediction accuracy for $k = 100$. For all but three of the experiments in Figure 4.8, there is little difference between prediction accuracy for training set sizes of 1, 5, 10, and 100. This is because there is little variation in the throughput observed during the experiments. A path with little or no variation in observed throughput over the course of an experimental run is the easy case for both SVR and HB throughput predictors, so these experiments will not be discussed any further in this paper. For three experiments in Figure 4.8, Amsterdam-London, London-Utah-1, and Utah-Cornell, the prediction accuracy for k values of 1, 5, and 10 is significantly lower compared to that for $k = 100$. The reason for this poor accuracy will be discussed in detail in Section 4.4.3.

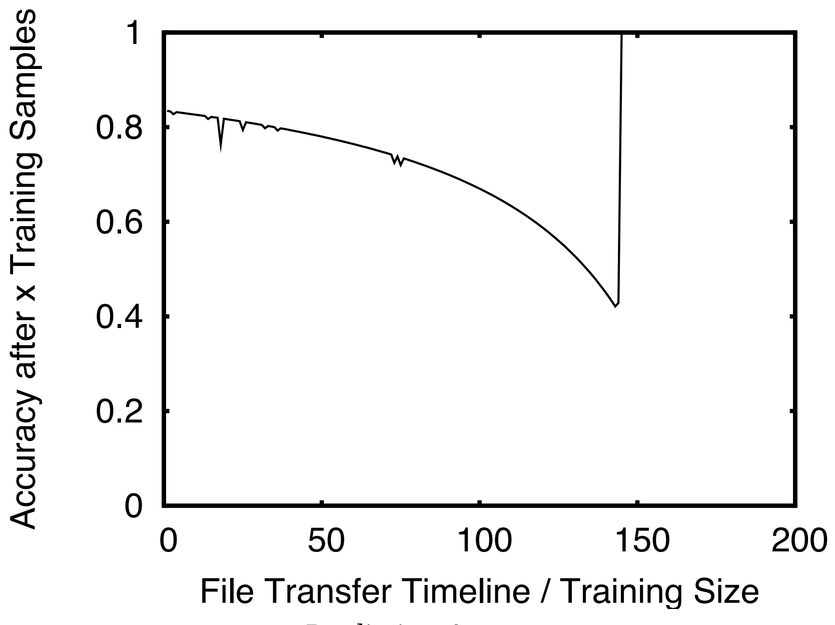
4.4.3 Detailed Analysis of Wide Area Results

In this section we analyze the reasons for poor SVR prediction accuracy for 5 paths from Figure 4.7 (Cornell-Amsterdam, London-Maryland-1, London-Utah-2, London-Utah-5, and NYU-Amsterdam) and 3 paths from Figure 4.8 (Amsterdam-London, London-Utah-1, and Utah-Cornell). We find that there are two dominant reasons for poor SVR prediction accuracy, background traffic level shifts and changes in background traffic on small timescales. We find that retraining can improve prediction accuracy for level shifts but not for changes in network conditions on small timescales.

In the analysis that follows, we use a pair of graphs per experiment to illustrate the details of each experiment. Consider Figures 4.9a and 4.9b for London-Utah-5. The first graph, the *throughput profile*, is a time series representation of the actual throughput observed during the experiment. The second graph, *prediction accuracy*, is a more detailed version of the bar graphs of Figure 4.8, incorporating all possible values of k. At an x value of k, the first k of the 200 samples are used as the training set, and the remaining $200 - k$ samples as the test set. The y value is the



(a) Throughput Profile



(b) Prediction Accuracy

Figure 4.9: London-Utah-5: an example of a wide area path with level shifts. Time on the x-axis is represented in terms of file transfer number

fraction of samples in the test set of $200 - k$ whose predicted throughput is within 10% of the actual throughput. As the value of k increases, the training set becomes larger compared to the test set, because the total number of samples is fixed at 200. For values of k close to 200, the test set is very small, and thus the prediction accuracy values, whether they are high or low, are not as meaningful.

4.4.3.1 Level Shifts

Level shifts were responsible for poor prediction accuracy in four experiments: Utah-Cornell, London-Utah, London-Utah-2, and London-Utah-5. We discuss London-Utah-5 in detail as a representative example.

Figures 4.9a and 4.9b are the *throughput profile* and *prediction accuracy* graphs for London-Utah-5. Figure 4.9a shows that a level shift from a throughput of 1.4 Mbps to 0.6 Mbps occurs after 144 file transfers, and a level shift from 0.6 Mbps back to 1.4 Mbps occurs after 176 transfers. Figure 4.9b shows that prediction accuracy decreases from 0.84 at 1 file transfer to a global minimum of 0.40 at 144 file transfers, and then increases very rapidly to 1.00 at 145 file transfers. This result can be explained by the fact that the level shift is not included in the training data. Thus, the prediction function will generate samples at the first throughput level accurately, and those at the second throughput level inaccurately. Prediction accuracy decreases from 1 to 144 because there are relatively fewer samples at the first level and more at the second level in the test set as the value of x increases, and thus an increasing fraction of inaccurate predictions, leading to a decreasing trend in prediction accuracy. If the division into training and test sets is at the level shift boundary, in this case the first 144 file transfers, the training set consists only of measurement samples before the level shift and the test set of samples after the level shift. All predictions will be inaccurate (assuming the level shift changes the throughput by more than 10%, our threshold) because the prediction function has never encountered the new throughput level. Hence, we observe minimum prediction accuracy at the level shift boundary, *i.e.,* 144 transfers. If the division into training and test set includes a level shift, some samples with the new network conditions and resultant new throughput are included in the training set. The prediction function is now aware of the new network conditions, and is able to make better predictions in the new conditions. In our example, including only *one* sample from after the level shift in the training set, the 145th sample, is sufficient to allow all throughputs at the lower levels to be predicted accurately. That is, the SVR predictor needs the minimum possible training set size (one single sample) for the new network conditions before it can generate accurate predictions.

Figures 4.10a and 4.10b compare the behavior of SVR and HB predictors for a level shift. The training set for SVR consisted of the first 145 samples, *i.e.,* 144 samples at the first throughput level and 1 sample at the second throughput level. The test set consisted of the remaining 55 samples. For HB, recall that there is no separation into training and test sets, and retraining occurs after every measurement sample. Comparing Figures 4.10a (SVR) and 4.10b (HB), we see that the SVR predicted throughput follows the actual throughput very closely, while the HB predicted throughput takes some time to catch up with actual throughput after a level shift. If the SVR predictor has knowledge of the level shift, its prediction accuracy is much better than that of HB. After the second level shift (176 samples) no further training of the SVR predictor is required to predict the remaining 23 correctly. The predicted throughput in Figure 4.10a follows the actual throughput closely at the level shift after 176 transfers even though the training set consists of only the first 146 samples.

The above example shows that the SVR predictor has two advantages over the HB predictor. First, it can adapt instantaneously, *i.e.*, after a single training sample, to a level shift, while the HB predictor takes longer. Second, it shows that unlike the HB predictor, the SVR predictor needs to be trained only once for a given set of conditions. The results for the other three wide area experiments that contained level shifts are similar to those of the London-Utah-5 experiment.

4.4.3.2 Changes Over Small Timescales

Changes in network conditions over small timescales reduced SVR prediction accuracy for Amsterdam-London, Cornell-Amsterdam, London-Maryland-1, and NYU-Amsterdam. We consider London-Maryland-1 as a representative example.

Figures 4.11a and 4.11b present the throughput profile and prediction accuracy of the London-Maryland-1 experiment. Figure 4.11a shows that until about the 60th file transfer, throughput is fairly steady around 2.7 Mbps, after which it starts to vary widely in the range between approximately 1.2 Mbps and 2.7 Mbps. Figure 4.11b shows that prediction accuracy is at a maximum of 65% at one file transfer, decreases in accuracy between 1 and 60 transfers, and varies between 50% and 60% between 60 and 180 transfers.

Unlike for level shifts, after the throughput profile changes and measurement samples of the new network conditions are included in the training set, prediction accuracy does not improve. Recall that we measure network conditions using BADABING for 30 seconds, and then transfer a file. In this experiment, after the 60th transfer, the network conditions vary so rapidly that they change between the BADABING measurement and the file transfer. Consequently, there is no consistent mapping between the measured network conditions and the transfer throughput, so a correct SVR predictor cannot be constructed, leading to poor prediction accuracy.

The HB predictor also performs poorly in these conditions. As shown in Figure 4.7, for 100 training samples for London-Maryland-1, HB has a prediction accuracy of 59% while SVR has an accuracy of 55%. When network conditions vary as rapidly as in the above example, it is not possible to predict throughput accurately using either SVR or HB because of the absence of a consistent relationship in network conditions just before and during a file transfer.

One difference we observed in experiments with level shifts versus experiments with throughput changes on small timescales was that level shifts occurred under lossless network conditions while throughput changes on small timescales occurred under conditions of sustained loss. Thus if only the average queuing delay on a network path changes, we observe a level shift; if a network goes from a no-loss state to a lossy state, we observe throughput changes on small timescales. A possible explanation is that if the network is in a lossy state, a particular TCP flow may or may not experience loss. Since TCP backs off aggressively after a loss, flows that do experience loss will have significantly lower throughput compared to those that do not experience loss, leading to large differences in throughput of flows. However, if only average queuing delay on a path changes, every flow on the path (unless it is extremely short) experiences the change in queuing delay, leading to a change in throughput of all flows, *i.e.*, a level shift, on the path.

4.5 Practical Deployment

In this section, we provide an overview of the implementation of PATHPERF and address three key issues related to running PATHPERF in operational settings. PATHPERF is openly available for download at `http://wail.cs.wisc.edu/waildownload.py`.

The PATHPERF distribution has three main components. The first is BADABING-RT, which measures packet loss and queuing delay on a path. The second is the main PATHPERF module. It initiates active measurements of path properties and file transfers, and preprocesses measurements for SVM^{light}, the third component, which is a popular C-language implementation of Support Vector Machines.

The RT in BADABING-RT stands for *round-trip*, because we modify the original BADABING one-way loss measurement tool to reflect probe packets back from the receiver to the sender. We make this change for two reasons. First, the original BADABING implementation assumes that the clocks at the sending and receiving host are synchronized, an assumption that generally does not hold. Reflecting the probe back to the sender allows loss to be measured without the need for synchronized sender and receiver clocks. Also, probe reflection allows queuing delay to be measured without the introduction of any additional measurement traffic. The method for computing *loss frequency* and *loss duration*, the two loss metrics that BADABING reports, based on probe traces, remains unchanged. The main challenge in the use of BADABING-RT for PATHPERF was finding a set of default values for the BADABING-RT parameters α and τ (see [115] for detailed descriptions) that would work well over a diverse set of paths. This is important for making PATHPERF easy to use out-of-the-box. We were able to find such a set of default operating parameters by examining how parameter settings affect the computation of the *loss frequency* and *loss duration* values for a variety of laboratory traces. The default value of α is 0.0001, and that of τ is 0.005.

The main PATHPERF module is 2500 lines of C code. It performs several tasks. The first is to conduct BADABING-RT active measurements and file transfers based on user-specified input parameters such as the duration of the BADABING-RT run and the file transfer size distribution. The second is to preprocess the active measurements for use by SVM^{light}. This step involves *normalizing* each input feature by subtracting the mean of each feature from the respective feature values and then dividing by the standard deviation. Normalization places all input features on the same scale, thus preventing features with a larger original scale from biasing the solution. The third is to call SVM^{light} at the appropriate times in the experiment to construct or update the predictor and to generate predictions. The final is to maintain records of all steps, from the active measurements to the prediction generation. In cases where prediction accuracy is low, these records help the user to determine whether the low accuracy is due to excessive short-term variation in path conditions or poor choice of configuration parameters such as the retraining frequency.

We do not make any changes to the SVM^{light} package for use with PATHPERF, and it is included in the distribution in its original form. SVM^{light} constructs a predictor using a training set, and then generates predictions using test sets, as described in Section 4.2. The main decisions associated with using SVR are *(a)* the choice of input features, *(b)* the choice of kernel and *(c)* the choice of kernel parameters, if any. We use file size, queuing delay and loss as our input features, and find that the addition of available bandwidth as an input feature does not improve prediction accuracy, as described in Section 4.3. We use a Radial Basis Function (RBF) kernel because it is known to perform best in practice when the relationship between the input features and the target value is known to be non-linear. We computed the parameters for the RBF kernel using cross-validation on our laboratory data as described in Section 4.2.

Network Load Introduced by PATHPERF. Traffic introduced by active measurement tools is a concern because it can skew the network property being measured. Furthermore, network operators and engineers generally wish to minimize any impact measurement traffic may have on customer traffic.

For history-based TCP throughput estimation methods, the amount of traffic introduced depends on the measurement protocol. For example, two approaches may be taken. The first method is to periodically transfer fixed-size files, and the second is to allow a TCP connection to transfer data for a fixed time period. The second approach was taken in the history-based evaluation of He *et al.* [54]. To estimate the overhead of a history-based predictor using fixed-duration transfers, assume that (1) the TCP connection is not $rwnd$-limited, (2) that the fixed duration of data transfer is 50 seconds (as in [54]), (3) throughput measurements are initiated every 5 minutes, and (4) throughput closely matches available bandwidth. For this example, assume that the average available bandwidth for the duration of the experiment is approximately 50 Mb/s. Over a 30 minute period, nearly 2 GB in measurement traffic is produced, resulting in an average bandwidth of about 8.3 Mb/s.

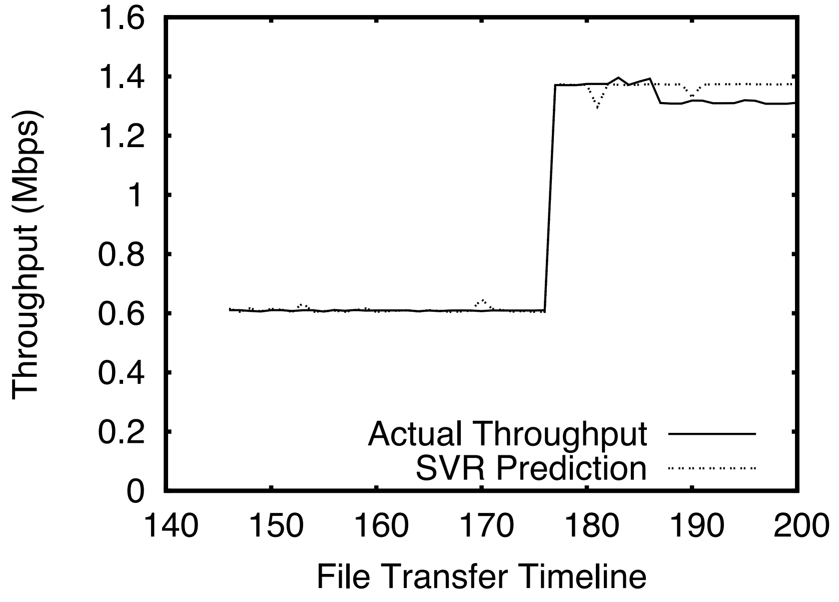
In the case of PATHPERF, measurement overhead in training consists of file transfers and queuing/loss probe measurements. In testing, overhead consists solely of loss measurements. Assume that we have a 15 minute training period followed by a 15 minute testing period. Assume that file sizes of 32 KB, 512 KB, and 8 MB are transferred during the training period, using 10 samples of each file, and that each file transfer is preceeded by a 30 second BADABING measurement. With a probe probability of 0.3, BADABING traffic for each measurement is about 1.5 MB. For testing, only BADABING measurements must be ongoing. Assume that a 30 second BADABING measurement is initiated every three minutes. Thus, over the 15 minute training period about 130 MB of measurement traffic is produced, resulting in an average bandwidth of about 1.2 Mb/s for the first 15 minutes. For the testing period, a total of 7.5 MB of measurement traffic is produced, resulting in a rate of about 66 Kb/s. Overall, PATHPERF produces 633 Kb/s on average over the 30 minute measurement period, dramatically different from a standard history-based measurement approach. Even if more conservative assumptions are made on the history-based approach, the differences in overhead are significant. Again, the reason for the dramatic savings is that once the SVR predictor has been trained, only lightweight measurements are required for accurate predictions.

Detecting Problems in Estimation. An important capability for throughput estimation in live deployments is to detect when there are significant estimation errors. Such errors could be indicative of a change in network routing, causing an abrupt change in delay, loss, and throughput. It could also signal a pathological network condition, such as an ongoing denial-of-service attack leading to endemic network loss along a path. On the other hand, it may simply be a measurement outlier with no network-based cause.

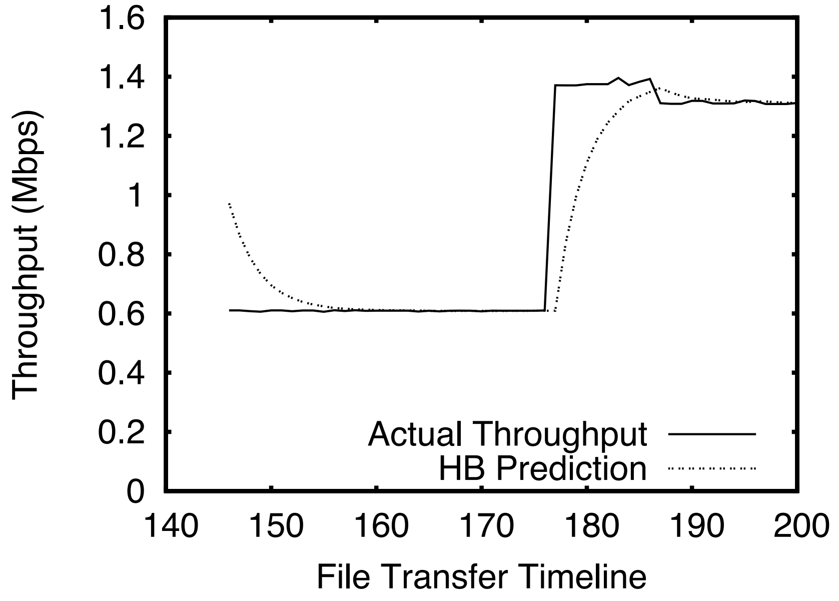
As discussed in Section 4.3.1.3, normality allows us to use standard statistical machinery to compute confidence intervals (*i.e.*, using measured variance of prediction error). We show that prediction errors are consistent with a normal distribution and further propose using confidence intervals as a mechanism for triggering retraining of the SVR as follows. Assume that we have trained the SVR predictor over n measurement periods (*i.e.*, we have n throughput samples and n samples of L and Q). Assume that we then collect k additional throughput samples, making predictions for each sample and recording the error. We therefore have k error samples between what was predicted and what was subsequently measured. Given a confidence level, *e.g.*, 95%, we can calculate confidence intervals on the sample error distribution. We can then, with low frequency, collect additional throughput samples to test whether the prediction error exceeds the interval bounds. (Note that these additional samples may be application traffic for which predictions are used.) If so, we may decide that retraining the SVR predictor is appropriate. A danger in triggering an immediate retraining is that such a policy may be too sensitive to outliers regardless of the confidence interval chosen. More generally, we can consider a threshold m of consecutive prediction errors that exceed the computed confidence interval bounds as a trigger for retraining.

Predictor Portability Thus far, since we have only considered predictors trained and tested on the same path, a number of features, such as the identity of a path and MTU, are implicit. This makes SVR simpler to use for path-specific prediction, because a lot of path details that would be required by other approaches, such as formula-based prediction, can safely be hidden from the user.

If two paths are similar in all features, implicit and explicit, then a predictor trained on one path using only Q and L (and possibly AB) can be used directly for the other path. If paths differ in implicit features, then those features will have to be added explicitly to the predictor to allow the predictor to be portable. Fortunately, SVR extends naturally to include new features. If a complete set of features effecting TCP throughput can be compiled, and there is a wide enough set of consistent training data available, then a *universal* SVR predictor that works on just about any wireline Internet path might be a possibility. A potential challenge here is identifying and measuring all factors – path properties, TCP flavors, operating system parameters – that might affect throughput. However, this challenge can be surmounted to quite an extent by active path measurements, using TCP header fields, and parsing operating systems configuration files to include operating parameters as SVR features.

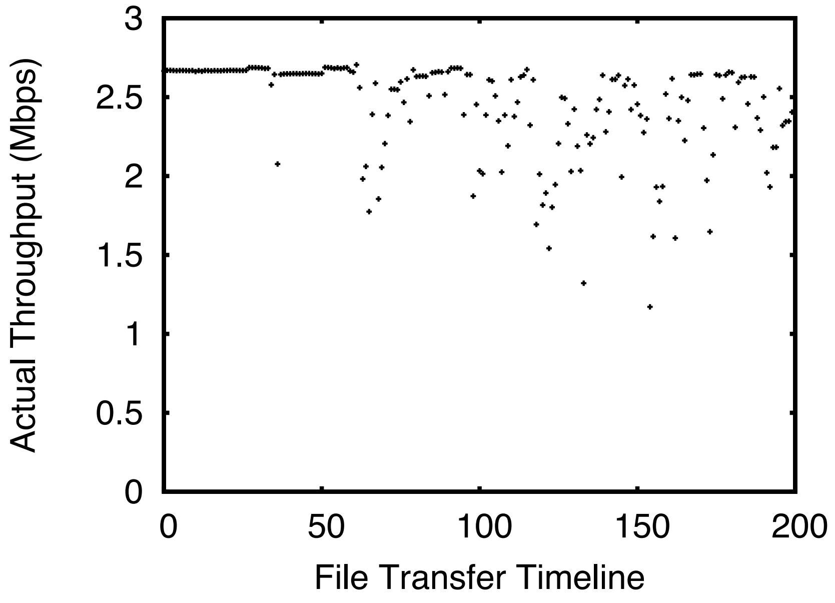


(a) SVR Prediction Accuracy

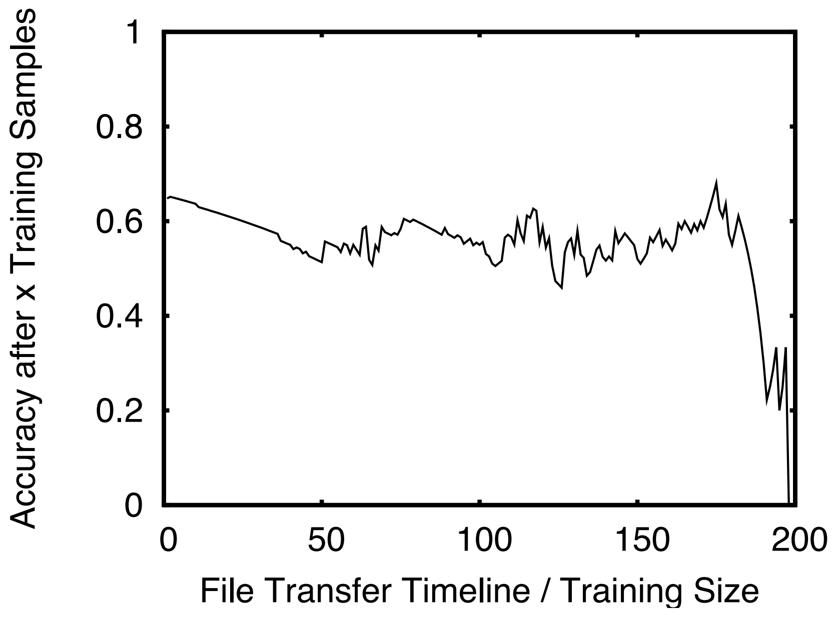


(b) HB Prediction Accuracy

Figure 4.10: London-Utah-5: Comparison of SVR and HB prediction accuracy for background traffic with level shifts. The x axis is the file transfer timeline starting at 145. The y axis is throughput.



(a) Throughput Profile



(b) Prediction Accuracy

Figure 4.11: London-Maryland-1: An example of a wide area path with throughput changes on small timescales

Chapter 5

Wireless Throughput Prediction

5.1 Overview

Over the past decade, there has been a great increase in the deployment of WiFi Access Points (APs) in homes, neighborhoods, and public areas. The increasing density of WiFi APs has enabled a new model of wireless network connectivity, where a guest user passing through the range of one or more APs can gain temporary Internet access. This type of access is commonly referred to as *opportunistic networking*. While the users of these opportunistic networks can be either stationary or mobile and vehicular, many research efforts in the past five years have focused on developing protocols and applications for vehicular opportunistic networks (Chapter 2 Section 2.2.2).

A forecast of the throughput an application can expect in a given opportunistic wireless environment can be quite valuable. It can help adaptive applications adjust their parameters according to prevailing network conditions. It can tell the guest user whether bandwidth-intensive and time-sensitive applications such as voice over IP and live video will be able to operate successfully in a given network environment. In dense wireless environments, knowledge of expected throughput can allow clients to select the best AP, *i.e.*, the highest throughput AP, if more than one AP is within range. TCP throughput prediction can provide the guest user with the TCP-friendly rate, which is useful for limiting non-TCP traffic that the guest can introduce to levels that are not disruptive for the network. This is an important application, because bounding guest traffic to benign levels is essential to the success of opportunistic networks. Guest users have an incentive to respect this bound, because if they do not, it is unlikely that APs will allow the open access required for opportunistic networking, and the opportunistic networking paradigm will fail.

Throughput prediction for wireless networks is more challenging than throughput prediction for wireline networks. The wireless network environment is much more complex at the physical (PHY) and medium access control (MAC) layers than the wireline environment. In wireline networks, packet loss is mainly due to congestion and consequent buffer overflow at routers. In wireless networks, the dominant types of losses are radio losses and interference losses. Radio losses occur due to high bit error rates (BER) caused by poor channel quality. Channel quality is determined by the signal-to-noise ratio (SNR) of the link. If the signal strength is high compared to the background noise, the receiver can decode the transmission successfully with high probability. If, however, the signal is weak, *i.e.*, its strength is similar to the background noise level, the BER increases, so the probability of successfully decoding the signal decreases, resulting in a higher packet loss rate. Interference losses occur when multiple sources within range of each other transmit at the same time in the same frequency band. The presence of these PHY and MAC layer phenomena, and of mechanisms for mitigating loss at the PHY and MAC layer such as rate adaptation and RTS-CTS, in addition to congestion and transport protocol behavior, means that a large number of factors effect wireless throughput in complicated ways, making the task of accurately predicting throughput very difficult. The difference in the wireline and wireless environments means that the techniques developed for wireline throughput prediction cannot be applied to the wireless environment because the assumptions on which wireline path measurement tools are based do not hold for the wireless environment. The presence of mobile vehicular clients makes throughput prediction still more difficult for two reasons. First, the network environment becomes more dynamic and variable and hence less predictable. Second, the throughput prediction has to be generated very quickly to be useful because a vehicular client is within the range of any given AP for only a short period of time.

As discussed in Chapter 1 Section 1.1, network performance studies aim to maximize analysis accuracy while dealing with the often competing requirements of analysis comprehensiveness, extensibility and maintenance, robustness, agility, and practicality. For 802.11 capacity modeling and throughput prediction efforts to date, such as [47, 97, 105, 103, 68, 92, 80], analysis comprehensiveness and practicality have been the primary challenges.

Analysis comprehensiveness has been a major challenge for wireless throughput modeling and prediction efforts because the complexity of wireless environment has prevented the modeling of fully general wireless networks. The earliest of the above-listed efforts [47] considered only interference-free broadcast traffic. Subsequent studies also made several simplifying assumptions, such as considering only binary interference instead of an arbitrary number of interference sources, modeling the use of a single 802.11 transmission rate instead of the presence of rate adaptation algorithms, and assuming constant bit rate traffic instead of handling the presence of a transport layer protocol such as TCP. Analysis comprehensiveness increased with each model, with the latest and most complete, [80], handling an arbitary number of sources of interference interference, unicast traffic, and TCP. Hence analysis comprehensiveness in the prior work has been achieved via a laborious incremental process.

For the above-listed 802.11 throughput prediction techniques, the issue of practicality arises in the form of assumptions about (a) system opaqueness and (b) permissible measurement delay. System opaqueness is a challenge because the studies assume extensive knowledge of the network topology, such as the number of interfering sources, their traffic demands, and pairwise measurements of the quality of the channel between the interfering sources. This information cannot be gathered without the cooperation of all the nodes in the network. Since clients in real wireless networks are autonomous, their cooperation cannot be assumed. While minimizing the measurement delay required to generate throughput predictions is desirable in any scenario, it is essential for opportunistic vehicular networks because studies such as [40] show that at normal driving speeds most vehicles spend 10 seconds or less within range of an AP, so to be useful to an application, throughput prediction have to be generated in considerably less than 10 seconds. However, the throughput modeling and prediction techniques in prior work require pairwise measurements between all the nodes in the network, *i.e.*, require $O(n^2)$ measurements for an n-node network, a process that takes tens of seconds, so these techniques cannot be applied to vehicular wireless networks.

Our approach uses a set of short active probes to measure network path characteristics. We use Support Vector Regression (SVR) to construct a prediction based on the relationship between the active probe measurements and path throughput. Our measurements are conducted between the target client (the client that requests the throughput prediction) and the AP and do not involve any other nodes in the network, so they surmount the system opaqueness challenge. Our approach can generate predictions with useful accuracy using probes that are only 0.3 seconds long, so unlike prior work techniques our approach overcomes the measurement delay challenge and is rapid enough even for vehicular opportunistic networks. A major strength of our approach is the ease with which it handles analysis comprehensiveness. The combination of simple and short active measurements and SVR-based prediction allows our approach to predict TCP throughput for fully general wireless networks and does not require us to make any simplifying assumptions regarding the number of sources of interference in the network or the protocols and optimizations deployed. This is in contrast to the prior work, which had to go through a long process of building incrementally more complete models to achieve analysis comprehensiveness.

We consider the problem of TCP throughput prediction empirically by taking measurements of wireless client and AP interactions over a broad range of operating conditions including varying number of APs in the network, conducting experiments indoors and outdoors, with client stationary or moving at walking or driving speeds. In each case, third party interference is created by systems running load generators that emulate web-like, bursty cross traffic.The prediction accuracy of our method is cause for cautious optimism. We find that for all wireless environment configurations, wireless throughput prediction accuracy is lower than the accuracy of our wireline SVR-based predictor (Chapter 4, which is the most accurate wireline throughput predictor in the literature. We find in most of our experiments that only 10% to 30% of predictions are within 10% of actual throughput. In comparison, for our SVR-based wireline predictor, 50% or more of predictions are within 10% of actual: this is a factor of 2 to 5 better than wireless prediction accuracy. However, we also find that 80% to 100% of predictions are within a factor of two of actual throughput. The factor of two bound on accuracy means that predictions are useful for certain (but not all) applications. A major strength of our approach is that it can be used in the vehicular scenario because it can achieve this bound on accuracy using measurements lasting for as little as 0.3 seconds, and because this bound holds even when nodes are driving at speeds of 15-25 mph.

To gain some insight into the disparity between wireline and wireless TCP throughput prediction accuracy, we compare the SVR-based predictor with a predictor based on Exponentially Weighted Moving Averages (EWMA), a standard time series forecast. We find that the EWMA-based predictor has accuracy similar to the SVR-based predictor. [1]. The fact that both predictors have similar accuracy suggests that greater inherent variability in wireless

[1] In spite of comparable accuracy between the two, the SVR-based predictor remains the predictor of choice because the EWMA-based predictor cannot generate predictions quite as rapidly, as discussed in Section 5.4

environments is responsible for the disparity in achievable wireline and wireless TCP throughput prediction accuracy. We conclude by providing evidence in support of this hypothesis.

5.2 Experimental Setup

In this section we present our indoor (laboratory) and outdoor (drive-by) experimental environments, and the measurements we conducted in these environments to construct a wireless throughput predictor.

5.2.1 Experimental Environment

We use 802.11a for our experiments: this allows us to avoid interference with our department's and other departments' 802.11b/g networks. We use 802.11a channel 36 for all our experiments.

Figure 5.1 illustrates the experimental setup for our laboratory experiments. There are four primary components to the setup: the wireline nodes, the wireless nodes, two commodity access points (APs), and a monitor node.

The wireline and wireless nodes are connected in a dumbbell topology via the APs and a switch. The APs are Cisco AP1200, running IOS version 12.3(8), with single Rubber Duck antenna, and integrated 802.11a module/antenna. The switch is a commodity LinkSys 10/100 16-port Workgroup Hub. The wireline nodes and switch are connected to the AP via 100Mbps Ethernet connections. The maximum data rate for 802.11a is 54Mbps; having 100 Mbps wireline links insures that the wireless, rather than the wireline, part of the network is the throughput bottleneck.

The wireline nodes are identically configured Sun 4200 AMD Opteron 275 (dual Core) nodes, with 4 GB RAM, Intel 82546EB (e1000) chips, running CentOS 5.0. The wireless nodes are MacBooks and MacBook Pros with default vendor configurations. For the wireless network interfaces, for both indoor and outdoor experiments, the defaults were *(a)* no RTS/CTS, *(b)* rate adaptation enabled, and *(c)* no mac layer fragmentation. We used the default network interface configurations because in production opportunistic networks the client nodes will to be heterogeneous and independently configured by users and/or vendors.

The monitor node is located next to the foreground wireless node for all experiments. This node is a Dell 700M, 1.6Ghz P4, with a Cisco CB21AG 802.11a/b/g cardbus adapter. The monitor node does not transmit any packets, but captures all the packets it sees on a specified channel. The traces thus collected are used for throughput prediction.

Figure 5.2 illustrates the setup for the drive-by experiments. The outdoor setup is different from the indoor setup in three ways. First, we replaced the Sun 4200 AMD Opteron nodes (the wired nodes) with Powerbook G4 and MacBook Pros because the Sun nodes were rack-mounted machines and hence not portable. Second, instead of using a MacBook Pro as the foreground wireless node, we used a 1 Ghz VIA Nehemiah node, with 512MB RAM, Netgear WAG311 with integrated 5 DBi antenna, running Gentoo 1.12.9, which is a 2.6.20.7 Linux kernel. The antenna extended the range of the wireless signal sufficiently to allow us to communicate with the AP while driving to a distance of approximately 400 ft in each direction. Third, we mounted the AP 20 feet above the ground to stay within line of sight of the car for a greater distance.

We used a portable generator to power our stationary nodes, which we placed halfway along our driving path, which was between two parking lots. The wireless foreground node and the monitoring node were placed in the car; the wireless foreground node's 5 DBi antenna was placed on the roof of the car to prevent the body of the car from lowering the signal strength.

We ran two sets of indoor experiments. In the first set, the foreground wireless node was stationary; in the second set, the foreground node was moved around in the hallway next to our lab at walking speed over a distance of 100 ft. The position of the foreground node in both these scenarios is illustrated in Figure 5.1. How far we could move the foreground node was restricted by the layout of the hallway.

For the outdoor experiments, we drove along a stretch of road between two parking lots that were approx. 850 ft apart at speeds between 15 and 25 mph. An athletic field bordered the road on one side; the other side was bordered by a residential area. We chose this particular location because its low vehicular traffic volume allowed us to maintain driving speeds for 15-25 mph, and having an open field on one side allowed us to maintain line of sight, and hence association with the AP, for approximately 400 ft on either side, compared to approximately 50 ft on either size in the indoor experiments. We limit the driving distance from the AP such that we always stay associated with the AP. When we experimented with driving far enough to get out of range and then coming back into range, the re-association time was high: we spent over 80% of our driving time trying to associate, because the client was scanning channels on all APs it was receiving beacons from. For vehicular networking to be feasible, significantly faster association techniques will be required. Developing fast association techniques is beyond the scope of this paper, but [40] has presented ways to speed up association for vehicular opportunistic networks.

5.2.2 Experimental Protocol

In our experiments, communication between wireless and wireline nodes is pairwise, *i.e.*, during an experiment, each wireless node sends data to a single, pre-assigned, wireline node and vice versa. One wireline-wireless pair is designated the foreground pair: we predict throughput from the perspective of this node. The other two node pairs generate background traffic.

We experiment with three different data flow scenarios. First, we download data from the wireline nodes to the wireless nodes: in this case, the principle direction of TCP data flow is from the AP to the wireless nodes, and only TCP ACKs travel in the reverse direction. This is how wireless clients are typically used. Second, we upload data from the wireless nodes to the wireline nodes. This maximizes contention for the wireless medium in a single-AP wireless network. In the upload case, there are three nodes, the wireless nodes, transmitting, while in the download case only the access point is transmitting for the most part, so in the upload case there is greater contention for the medium. Third, we use the 2-AP setup. Here, two APs are within range of each other, so even in the download case, there is contention for the medium. For our indoor experiments, in the 2-AP setup, the background traffic nodes are associated with the AP inside the lab (AP1), and the foreground node is associated with the AP in the hallway (AP2), which is mounted on the ceiling of the hallway (Figure 5.1). In the single AP upload and download cases, AP2 is turned off and all nodes are associated with AP1.

We use Harpoon [114], a flow-level network traffic generator, to generate background traffic. The traffic is open loop on/off TCP traffic, with file sizes drawn from a Pareto distribution and the time between transfers drawn from an exponential distribution, a pattern that closely resembles actual Internet traffic [36]. Each background node pair runs 0-6 data transfer threads at any given time: this represents a user doing a few different things on a laptop simultaneously, such as browsing the web and downloading software. We set the maximum number of threads per background node to 6. This varies the background traffic enough to make prediction non-trivial without overloading the wireless network to the point of congestion collapse and near-zero throughput.

Figure 5.3 presents the active probes we run on the foreground nodes. We run a series of Iperf transfers *t* seconds long separated by *w* seconds of wait time. Since our focus is on predicting throughput quickly, we use *t* and *w* values of 0.31, 0.42, 0.63, and 1.25 seconds. However, we also experiment with larger values for comparison. Background traffic runs continuously, during both the transfer phase and the wait phase of the foreground traffic.

5.3 Building a Robust Predictor

In this section, we describe how we construct wireless TCP throughput predictors using active measurements from Section 5.2.2. We then present an extensive evaluation of the accuracy of these predictors.

5.3.1 Predictor Construction

To construct the SVR-based predictor, we use a a Radial Basis Function (RBF) kernel (Chapter 3, Section 3.5) because RBF kernels tend to have the best accuracy in practice. We set the RBF kernel parameter γ for different series of experiments using cross-validation. We set the parameter C from Equation 3.34 using cross-validation as well. We use the SVM^{light} package [63] to construct our SVR-based predictor.

For comparison with SVR-based accuracy, we used the exponentially weighted moving average (EWMA), a time series-based predictor, stated below. We used an α value of 0.3.

$$\hat{R}_{i+1} = \alpha R_i + (1 - \alpha)\hat{R}_i$$

For each of the values 0.31, 0.42, 0.63, and 1.25 seconds of t and w for our active probes (Section 5.2.2), we collect 200 measurement samples. For EWMA, the moving average of the throughputs of the first *k* transfers serves as the prediction for the *(k+1)th* transfer. For SVR, the training set is every set of *k* consecutive transfers and the test set is the *(k+1)th* transfer. So, given the 200 training/test sample sequence, the first training set is made of samples 1 to *k* and the test set of sample *k+1*, the second training set of samples 2 to *k+1* and the test set of sample *k+2*, and so on. We use $k = 100$ for our analysis.

We use two schemes for SVR, called *SVR-InterTransfer* and *SVR-IntraTransfer*, as illustrated in Figure 5.3. In *SVR-InterTransfer*, the input feature to the SVR is the throughput of the *kth* transfer, and the target, *i.e.*, the value to be learned, is the throughput of the *(k+1)th* transfer. In *SVR-IntraTransfer*, the throughput of the first half of a transfer is the input feature, and the throughput of the second half is the target value. We divide the transfer into halves in terms of time elapsed rather than bytes transferred.

We use the metric *relative prediction error E* introduced in [54] to evaluate the accuracy of an individual throughput prediction. We denote the actual throughput by *R* and the predicted throughput by $\hat{R}$. *Relative prediction error* is defined as

$$E = \frac{\hat{R} - R}{min(\hat{R}, R)}$$

Throughout this paper, we use the distribution of the absolute value of *E* to report our results.

5.3.2 Evaluation of Prediction Accuracy

Figures 5.4–5.6 present the results of our experiments. We obtain the relative error of an individual prediction using the relative error metric presented in Section 5.3.1. We present our results as cumulative distributions of relative error. The x-axis of each graph is the relative prediction error, and the y-axis is the fraction of predictions with relative error of x or lower. A higher y value for a given x value means better prediction accuracy than a lower y value.

We begin by looking at the general trends in Figures 5.4–5.6, and then look at error distribution values in detail.

We first consider the error distribution for each individual graph. Different lines in a given graph represent combinations of data flow setups (downstream, downstream_2AP, upstream), and the length of the TCP transfer (0.31, 0.42, 0.63, or 1.25 seconds). [2] The overall trend is that, in each graph, all lines are clustered relatively close together. This means that at a given speed and for a given predictor (*EWMA*, *SVR-InterTransfer*, or *SVR-IntraTransfer*), traffic direction, number of APs, and transfer duration have little impact on prediction accuracy.

Next, we compare the different graphs in two ways: the speed of the wireless client (stationary, walking, or driving), and the predictor used.

The three graphs in each individual figure 5.4–5.6 represents increased client speed for a given predictor. The general trend in each figure as we go from (a) to (c), *i.e.*, as client speed increases, is that the gradient of the lines becomes less steep, *i.e.*, prediction accuracy decreases. This is consistent with intuition: as the client speed increases, the wireless environment becomes more dynamic, so predicting throughput becomes more difficult.

Next we consider the accuracy of different predictors for fixed client speeds, *i.e.*, we compare graphs (a) in Figures 5.4–5.6 with each other, graphs (b) with each other, and so on. The general trend in this grouping is that there is little difference in the gradient of lines, *i.e.*, prediction accuracy does not change significantly with a change in

[2]We present results for only 0.63 and 1.25 second transfers for the stationary and walking case to limit the number of graphs and maintain clarity; results for 0.31 and 0.42 second transfers were similar. We present only upstream traffic results for drive-by experiments because our downstream traffic traces turned out to be corrupted.

predictors. This observation is surprising: we had expected SVR-based predictors to do better than EWMA because they are quicker to respond to level shifts in background traffic. We also expected *SVR-IntraTransfer* to perform better than *SVR-InterTransfer*, because it has more current information about network conditions. One possible reason for *SVR-IntraTransfer* doing no better than *SVR-InterTransfer* could be TCP slow-start having a large impact on throughput because our TCP transfers are short, 1.25 seconds or less. To investigate this possibility, we experimented with larger timescales to minimize the effect of slow-start on throughput. Specifically, we used 8MB transfers separated by 15 second wait times. The results are presented in Figure 5.7. Once again, we observe that *SVR-IntraTransfer* does no better than *SVR-InterTransfer*, so we can rule out slow-start effects as being the reason why the relative prediction accuracy of *SVR-InterTransfer* and *SVR-IntraTransfer* is contrary to expectation.

Next, we consider the absolute prediction accuracy that is achieved. Here, there is cause for cautious optimism.

The prediction accuracy for wireless opportunistic networks using our methods is not comparable to the high prediction accuracy for wireline networks using the SVR-based predictor from Chapter 4. For the opportunistic wireless case, typically, 10% to 30% of predictions are within 10% of actual throughput, and 30% to 70% of predictions are within 20% of actual throughput. In comparison, for the wireline case using SVR, 50% or more of predictions are within 10% of actual [86], which is a factor of 2 to 5 better than the results we obtain for the wireless case.

However, in all cases, 80% to 100% of predictions are within a factor of two of the actual throughput. This bound on prediction accuracy means that predictions can still be useful for some purposes, such as TCP-friendly rate calculation. Applications wishing to use the TCP-friendly rate can be conservative and assume a factor of two over-prediction, and still be able to get a reasonable throughput without overloading the network. The alternative would be knowing only that available bandwidth is somewhere between 1 and 54 Mbps in an 802.11a or g network, which is a factor of 54, so a factor of two bound is considerably more useful to an application. Also, the prediction can be made with a measurement lasting only 0.3 seconds, which is essential if the node is mobile, because *(a)* the factor of two bound holds even for nodes driving at speeds of 15-25 mph, and *(b)* the prediction time is short enough to allow almost all the time within range of an AP can be devoted to transferring application data. On the other hand,predictions are not useful for applications such as highest-throughput access point selection, because the difference in achievable throughput between candidate APs is generally less than a factor of two (see, for example, [91]).

Next, we try to understand why it is difficult to achieve prediction accuracy comparable to the wireline case. We look at how throughput varies with time in Figure 5.8, for both stationary and driving nodes. Time is represented on the x-axis in terms of transfer numbers, and the y-axis shows the actual throughput for a particular transfer number. The main difference between the two graphs in the figure is that, on average, throughput is higher for the stationary case compared to the driving case. This is to be expected, because the average distance between the node and the AP is greater in the driving case. What is similar about the two graphs is the high variability of TCP throughput over time, represented by the sharp peaks and valleys in both graphs. While there is observable decrease in prediction accuracy as the speed of a node increases (observe the decreasing gradient from graphs (a) to (c) for each individual

Figure 5.4–5.6) this decrease is not very large. For stationary nodes, 30% to 70% of predictions are typically within 20% of actual throughput, for walking nodes 30% to 60%, and for driving nodes 20% to 40%, so the decrease in accuracy due to movement is about a factor of 1.5. In comparison, the decrease in accuracy relative to wireline TCP throughput prediction is a factor of 2 to 5. Also, *SVR-IntraTransfer* does no better than *EWMA* or *SVR-InterTransfer*, even though it has up-to-date path information available to it. All these observations suggest that TCP throughput in the wireless environment is inherently variable, regardless of whether nodes are stationary or mobile.

5.4 Practical Deployment

There is currently no tool available for predicting TCP throughput for mobile wireless networks. The results presented in this chapter provide the basis for the construction of such a tool. In this section, we discuss additional practical deployment issues relating to the development of a mobile wireless TCP throughput prediction tool.

Given the time constraints in the wireless opportunistic scenario where most clients are within the range of an AP for 10 seconds or less [40], clients require that the prediction be generated as soon as possible, ideally under 1 second, so it can be of use to applications. To enable generating a prediction within the 0.31–1.25 second window that we have evaluated, we envision a scenario where the predictor is pre-computed and maintained by the AP. In this case, only one active measurement between the AP and target client is required to generate a prediction using the pre-computed predictor. The AP can acquire the training set needed to pre-compute a predictor using past history based on past prediction requests and passive measurements of application traffic. To improve accuracy, different predictors can be maintained based on the type of interface card, rate adaptation algorithm used, and other well-known parameters that vary between clients. Geographic information, such as the road and direction of travel of the vehicular client, can also be used to augment the predictors, because studies such as [83] have reported that the behavior of the RF environment at specific locations tends to be predictable.

The need for a pre-computed predictor to meet the time constraints of vehicular opportunistic networks means that only the SVR-based predictor can be used in practice, and the EWMA-based predictor cannot. EWMA requires temporally contiguous data, and needs at least a few training samples to generate accurate predictions. SVR does not require temporally contiguous data. The time constraints in vehicular opportunistic networks do not allow for the creation of a training set on-the-fly, hence making the use of EWMA impractical.

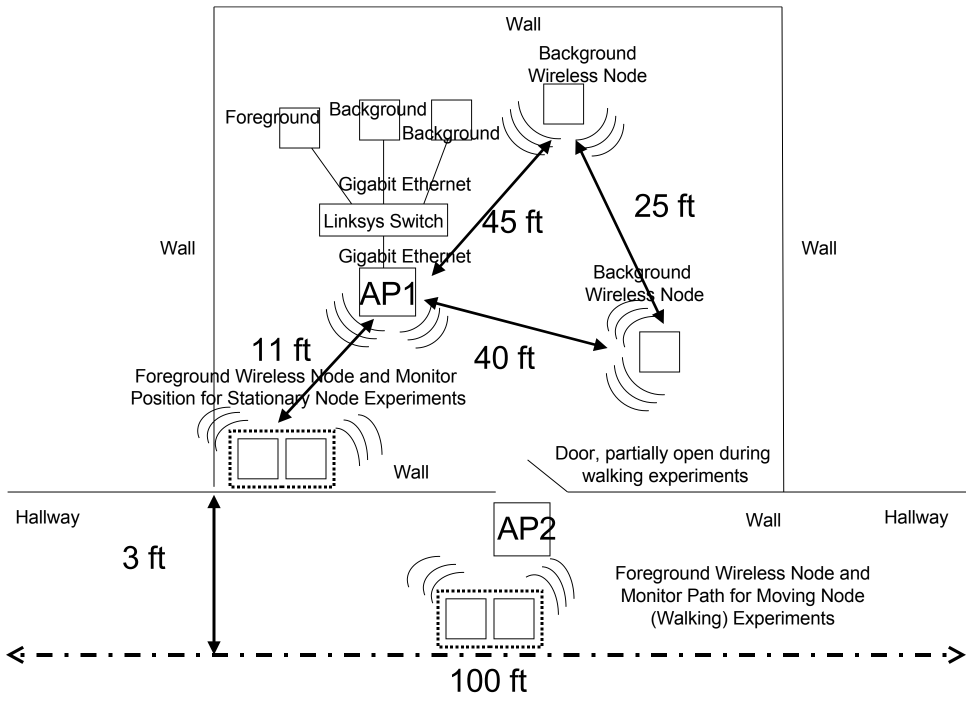

Wall
Background Wireless Node
Foreground
Background
Background
Gigabit Ethernet
Linksys Switch
Gigabit Ethernet
45 ft
25 ft
Wall
Wall
Background Wireless Node
AP1
11 ft
40 ft
Foreground Wireless Node and Monitor Position for Stationary Node Experiments
Door, partially open during walking experiments
Wall
Hallway
AP2
Wall
Hallway
3 ft
Foreground Wireless Node and Monitor Path for Moving Node (Walking) Experiments
100 ft

Figure 5.1: Laboratory (indoor) setup for stationary-node and walking-node experiments.

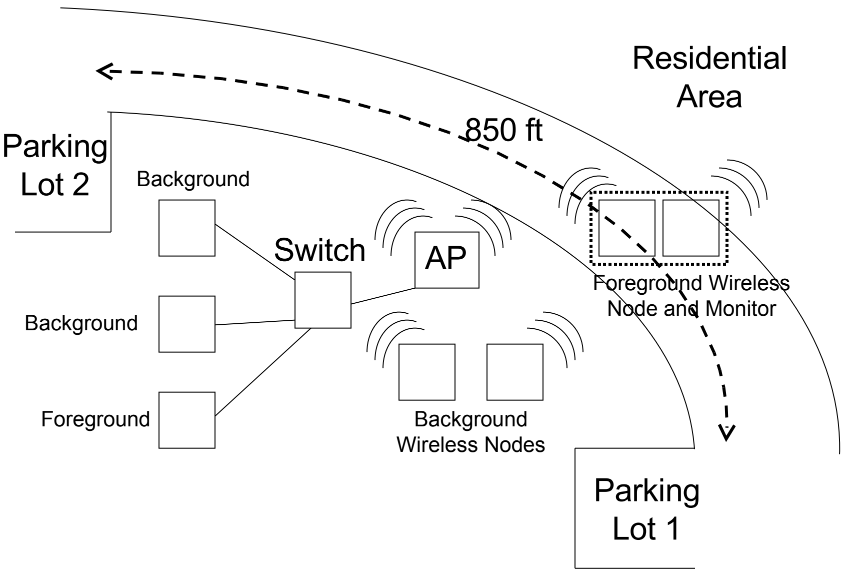


Figure 5.2: Setup for the drive-by experiments

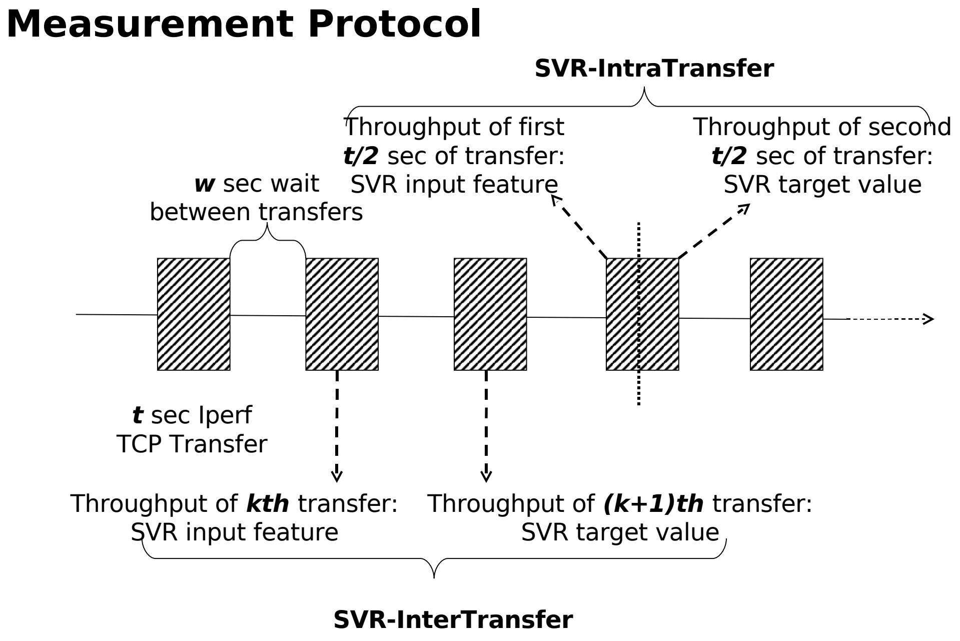
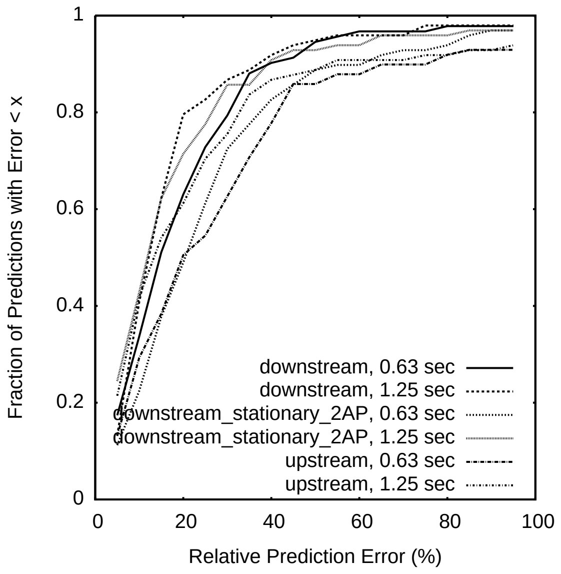
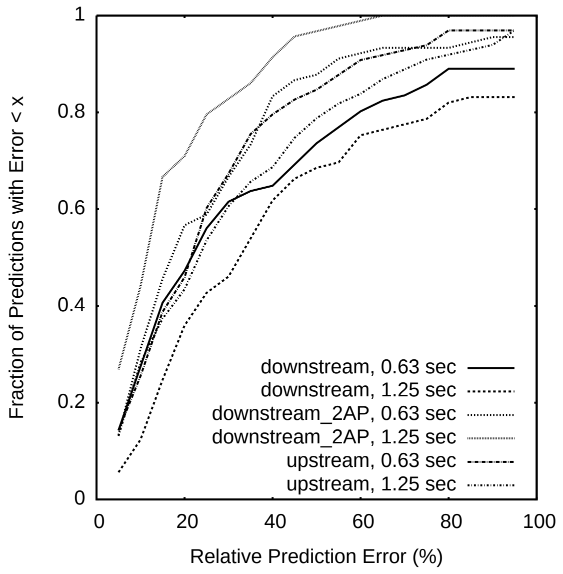

Measurement Protocol
SVR-IntraTransfer
Throughput of first t/2 sec of transfer: SVR input feature
Throughput of second t/2 sec of transfer: SVR target value
w sec wait between transfers
t sec Iperf TCP Transfer
Throughput of kth transfer: SVR input feature
Throughput of (k+1)th transfer: SVR target value
SVR-InterTransfer

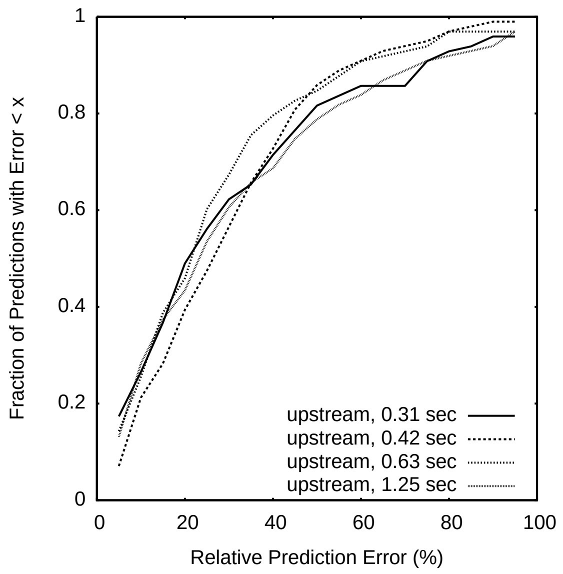
Figure 5.3: The foreground node measurement protocol used for all experiments.



(a) Stationary Node, EWMA

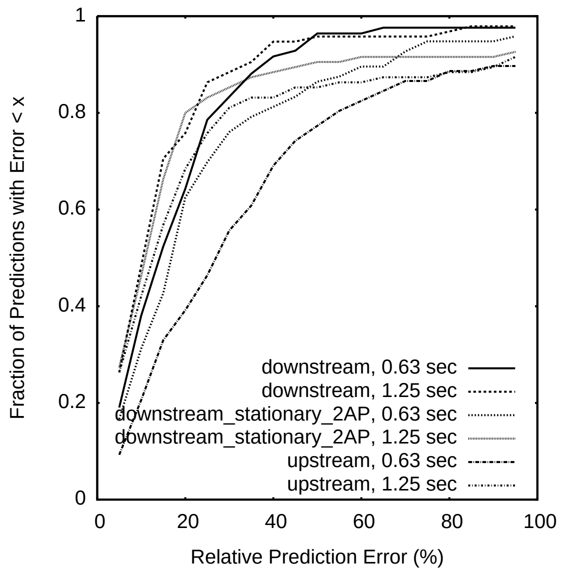


(b) Walking Node, EWMA

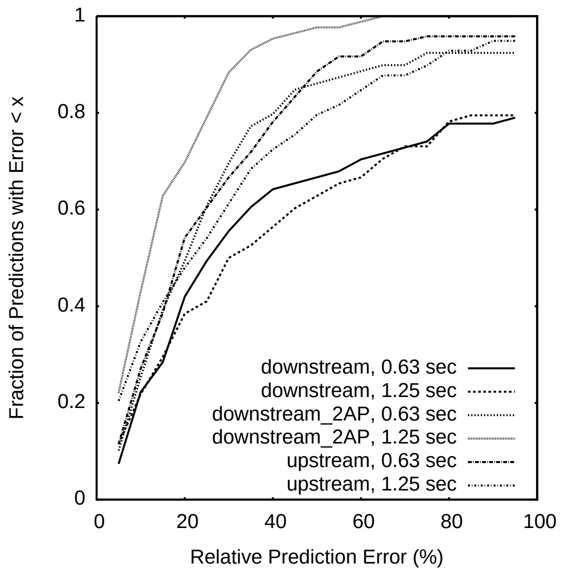


(c) Driving Node, EWMA

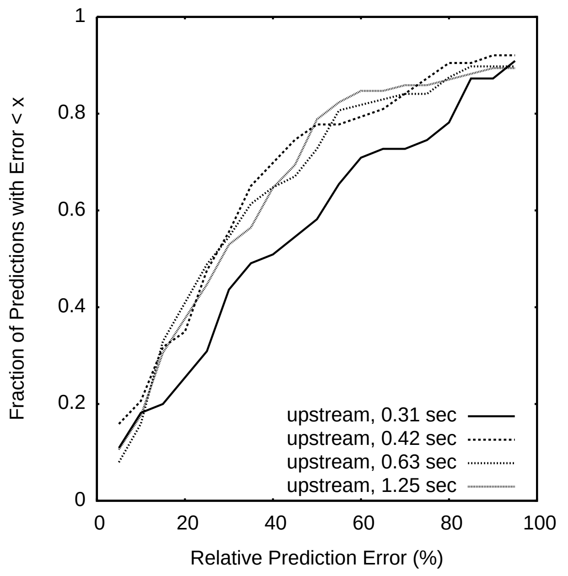
Figure 5.4: Prediction accuracy using the EWMA predictor for stationary, walking, and driving wireless nodes.



(a) Stationary Node, SVR-InterTransfer

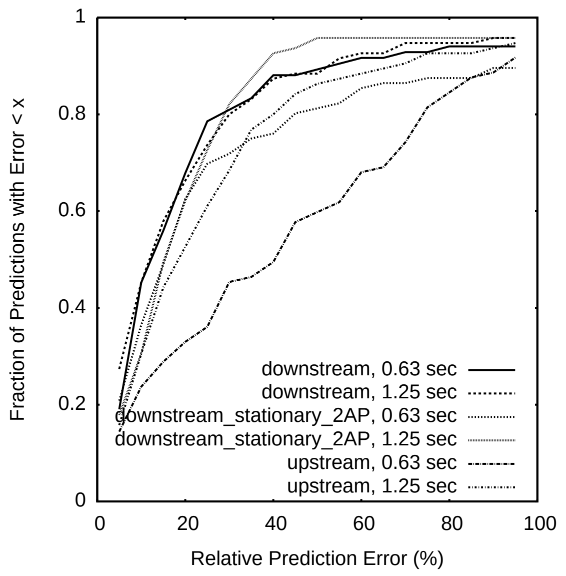


(b) Walking Node, SVR-InterTransfer

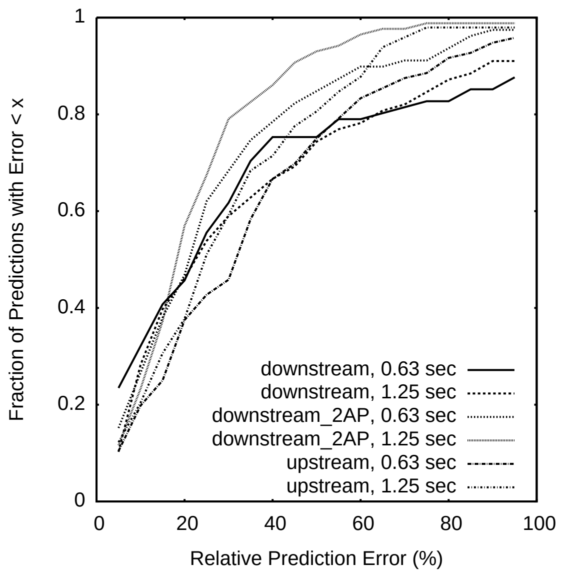


(c) Driving Node, SVR-InterTransfer

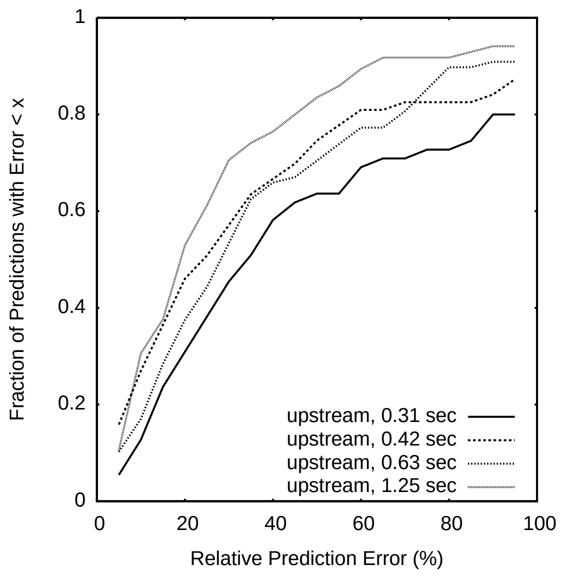
Figure 5.5: Prediction accuracy using the SVR-InterTransfer predictor for stationary, walking, and driving wireless nodes.



(a) Stationary Node, SVR-IntraTransfer

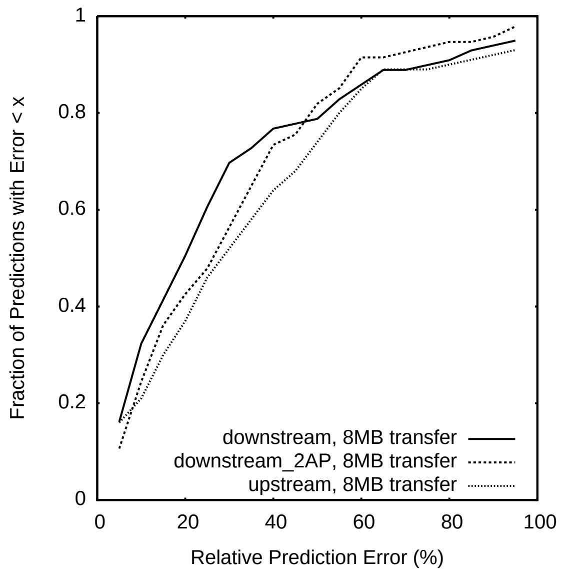


(b) Walking Node, SVR-IntraTransfer

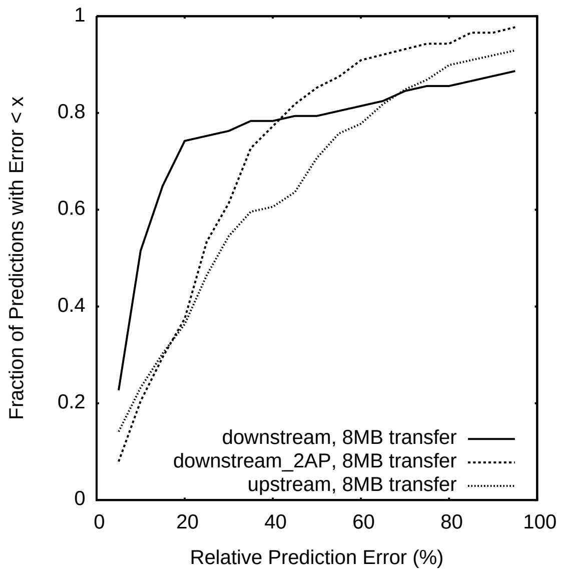


(c) Driving Node, SVR-IntraTransfer

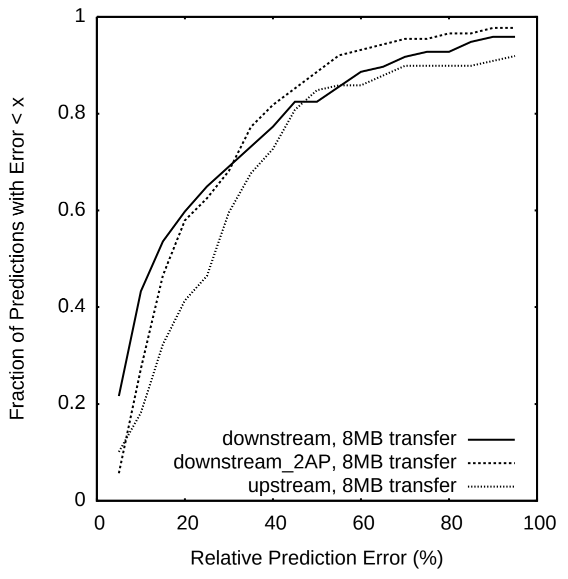
Figure 5.6: Prediction accuracy using EWMA and SVR predictors for stationary, walking, and driving wireless nodes.



(a) EWMA

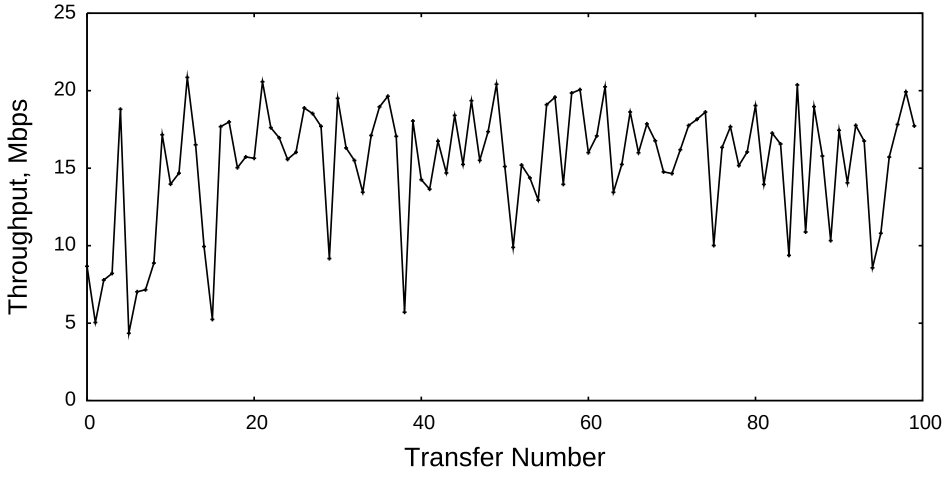


(b) SVR-InterTransfer

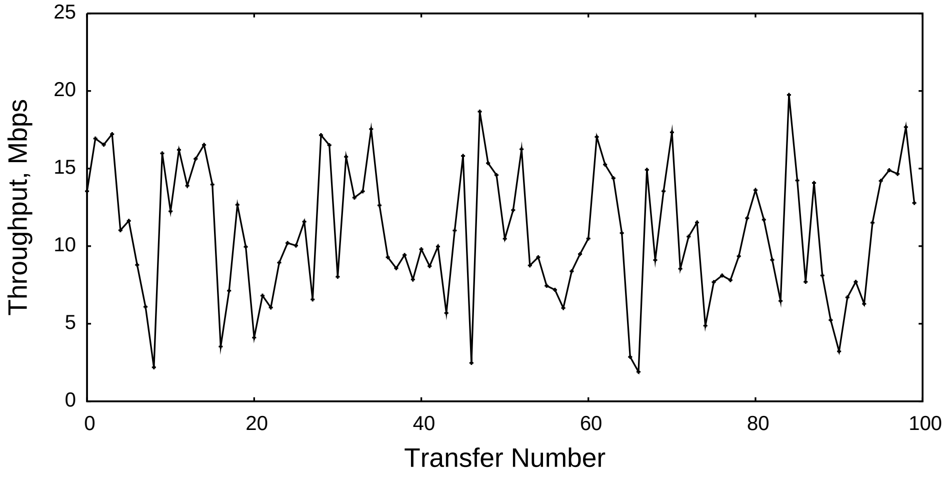


(c) SVR-IntraTransfer

Figure 5.7: Prediction accuracy using EWMA and SVR Predictors for Large (8MB) transfers from a Stationary Wireless Node.



(a) Stationary Node, Upstream Traffic, $t = 1.25$



(b) Driving Node, Upstream Traffic, $t = 1.25$

Figure 5.8: Throughput timeline for stationary and driving transfers, for upstream traffic with $t = 1.25$ sec

Chapter 6

Fingerprinting 802.11 Rate Adaptation Algorithms

6.1 Overview

802.11 supports multiple data transmission rates at the physical layer to allow senders to maximize throughput based on channel conditions. The modulation schemes used to encode data at lower rates are more robust to channel noise that those used for higher rates. If the channel quality is good, *i.e.*, the signal-to-noise ratio (SNR) is high, then higher data rates will maximize throughput because the bit-error rate (BER) will be low. If the channel is noisy, lower data rates will maximize throughput because the high BER at higher data rates will lead to increased loss and MAC-layer backoffs, resulting in poor throughput.

Designing algorithms that allow wireless senders to converge to the optimal rate for prevailing channel conditions in a timely fashion is challenging due to the difficulty of determining the cause of packet loss [104], limitations of the PHY/MAC interface in commodity wireless cards [129], and the assumption that a higher transmission rate always results in higher loss for a given RF environment not always being true [24]. Many attempts have been made to address this challenge, resulting in a large number of rate adaptation algorithms, with different algorithms performing best under different network conditions.

In this paper, we describe a method for identifying or *fingerprinting* the rate adaptation algorithms used in an 802.11 environment. We envision this capability as being part of a toolkit for automated performance analysis and debugging of production networks. The need for automated analysis and debugging has become increasingly urgent as 802.11 networks have grown to support large user populations. Client devices set their own configurations and connect and disconnect at will. Wireless network administrators have little control over, and knowledge of, network configurations, and cannot rely on cooperation from clients for performance analysis and debugging. Thus, practical performance analysis and debugging efforts for large-scale wireless networks such as [32, 82] are typically based entirely on passive monitoring, which requires no support or participation from clients. The presence of multiple rate adaptation algorithms introduces a new dimension of variability into 802.11 wireless networks. However, to the best of our knowledge, none of the passive monitoring-based performance analysis and debugging efforts in the literature consider the impact of 802.11 rate adaptation algorithms, despite the fact that the choice of rate adaptation algorithms can have a major impact on network throughput. The rate adaptation algorithm fingerprinting capability will provide additional information to passive monitoring systems to facilitate wireless network performance analysis.

We begin by investigating the details of the four open source rate adaptation algorithms from the popular MadWifi driver [9] that constitute the test cases for our study. Manual examination of implementations shows that the algorithms can result in many possible rate change permutations depending on the timing and pattern of packet transmissions and losses. The large space of permutations precludes a fingerprinting approach based on explicitly enumerating all possible cases of an algorithm's behavior and suggests the need for a learning-based approach for algorithm classification.

We develop a rate adaptation algorithm classifier based on Support Vector Machines using carefully selected input features from passive packet traces. We then conduct extensive experiments in a laboratory environment to identify combinations of features that result in the most accurate classification capability. Our results show that a classifier trained with a robust set of features can exhibit classification accuracy as high as 95%-100%. We show that careful selection of input features is necessary for achieving high classification accuracy. We also demonstrate that a classifier generated in one set of network conditions can identify algorithms in a different set of network conditions as long as the training data includes a sufficiently broad sampling of an algorithm's behavior.

Our method meets all the criteria of a useful performance analysis technique discussed in Chapter 1 Section 1.1. With a classification accuracy of 95%-100%, it comfortably meets the high accuracy criterion. It meets the analysis comprehensiveness requirement because we do not make any simplifying assumptions and evaluate our method under realistic network operating conditions. When a new rate adaptation algorithm needs to be added to the classification set, due to the use of SVM for classification, all that is required is the potential addition of input features, a task significantly easier than building a new explicit model to identify each new algorithm, so the method meets the extensibility and maintenance requirement. The discussion of the factors that effect classification accuracy in Section 6.4 illustrates that classification robustness can be achieved as long as the training set contains a comprehensive sampling of algorithm behavior in a wide range of conditions. The method is practical because it is based entirely on passive network traces and assumes no cooperation or active participation from the wireless clients in a network. The method is agile, *i.e.*, it can begin accurate classification in new or changing network environments without any delay, because pre-computed classifiers can be deployed, since we have demonstrated in Section 6.4 that classifiers are portable.

6.2 Rate Adaptation Algorithms

The development of our classifier is based on the four 802.11 rate adaptation algorithms implemented by the latest version (0.9.4) of the popular open-source MadWifi driver [9]. We chose an open source driver so we could gain insight into algorithm behavior and validate the results of the classifier based on manual algorithm inspection. In this section we summarize the MadWifi rate adaptation algorithms. Our objective is to highlight the complexities of an algorithm's behavior, in particular the fact that an algorithm's behavior can change dramatically depending on the prevailing network conditions. The need for a learning-based classifier arises because the large number of packet rate and retransmission patterns that can occur with a given algorithm would be very difficult to enumerate explicitly.

The MadWifi driver is designed for wireless cards using Atheros chips, which implement multi-rate retries [8]. The Hardware Abstraction Layer (HAL) exports a *retry chain*, consisting of 4 ordered pairs of *rate/count* values, *r0/c0* through *r3/c3*. The hardware makes *c0* attempts to transmit a given packet at rate *r0*, *c1* attempts to transmit a packet and rate *r1*, and so on. Once the packet is successfully transmitted, the remainder of the retry chain is discarded.

The rate adaptation algorithms have three tasks, *(a)* to select rate r and count c values for the retry chain, *(b)* to determine the conditions under which the retry chain values are updated, and *(c)* to determine how often to check for the update condition. In the remainder of this section, we outline how the four algorithms perform these tasks. We present the algorithms in increasing order of complexity.

Onoe [3] tries to maximize throughput by selecting the highest transmission rate that results in loss rate below a certain threshold. It uses a system of *credits* to decide whether to change the current rate *r0*. The credit associated with *r0* is increased by one if less than 10% of packets in the last interval need retries, and *r0* is increased to the next highest rate when the credit exceeds 10. The credit associated with *r0* is decreased if more than 10% of packets need retries. *r0* is decreased to the next lowest rate if the average number of retries per packet exceeds one. The interval for evaluating loss rate and updating credits is 0.5–1.0 seconds. *r1* and *r2* are set to the two rates consecutively below the current *r0*, and *r3* is set to the lowest possible rate (6 Mbps in 802.11a/g). *c0* is set to 4, and *c1, c2* and *c3* are set to 2. Since *r0* is updated indirectly based on credits, and the retry chain is 10 packets long, *Onoe* is rather slow to adapt to changing network conditions.

AMRR [74], like *Onoe*, tries to maximize throughput by selecting the highest transmission rate that results in loss rate below a certain threshold. If less than 10% of packets are lost in the last interval, the current *r0* is increased to the next highest rate, and if greater than 30% of packets are lost, it is decreased to the next lowest rate, otherwise it remains unchanged. Loss rate is evaluated every 10 packets. If a rate increase is attempted and it results in a loss, the interval for attempting the next increase is enlarged exponentially, up to a maximum of 50 packets. This is done to prevent unnecessary losses if the current transmission rate is the highest possible for the target loss rate. *r1* and *r2* are set to the two rates consecutively below the current *r0*, and *r3* is set to the lowest possible rate. *c0, c1, c2* and *c3* are all set to 1 to make the retry chain shorter and the algorithm more responsive compared to *Onoe*.

Sample Rate [24] selects *r0* by explicitly computing the rate most likely to maximize throughput in the prevailing network conditions, unlike *Onoe* and *AMRR*, which use the combination of loss minimization and rate maximization to estimate the best rate. *Onoe* and *AMRR* assume that a higher transmission rate will always result in a higher loss rate in a given environment. However, [24] shows this assumption to be incorrect, and shows that loss rate at a higher transmission rate may be lower depending on the modulation and encoding of the rates and the amount of noise in the RF environment. Motivated by these observations, *Sample Rate* explicitly computes throughput for a given rate based on the number of successful and failed transmissions and 802.11 parameters such as inter-frame spacing and

ACK transmission time. *Sample Rate* changes *r0* when another rate begins to yield better throughput. Since a rate other than the next highest or next lowest from the current *r0* may yield the best throughput, the algorithm has to periodically sample all other rates. 10% of transmission time is used for sampling alternate rates. Rates for sampling are selected intelligently, with rates more likely to improve throughput selected more frequently. *r0* is changed if sampling indicates that another rate will result in higher throughput. *r1* and *r2* are no longer set to the two next lowest rates after *r0*, rather they are set to rates with the next lowest throughputs. Throughput is reevaluated for the current *r0* and other candidate rates periodically and is smoothed using EWMA, with 5% of the weight coming from the last evaluation interval. *r0* is changed if another rate's EWMA throughput is greater.

Minstrel [2] is the most advanced rate adaptation algorithm implemented by the MadWifi driver. It improves on two aspects of *Sample Rate*. First, it sets *c0, c1, c2* and *c3* based on *r0, r1, r2* and *r3* such that the retry chain completes within 26 ms, a time limit selected to minimize TCP performance deterioration in case of losses. Second, it was noted that even with *Sample Rate's* intelligent selection, sampling alternate rates resulted in use of low rates and low throughput. *Minstrel* tries to avoid this problem by more sophisticated sampling rate selection, the complete details of which can be found in [2]. The throughput is calculated in a manner similar to *Sample Rate*. It is reevaluated for the current *r0* and other candidate rates every 100ms. The value is smoothed using EWMA, with 25% of the weight coming from the latest 100ms interval, and *r0* is changed if another rate's EWMA throughput is greater.

6.3 Experimental Setup

In this section we describe our experimental testbed and how we used it to generate packet traces for building the SVM-based 802.11 rate adaptation algorithm classifier.

6.3.1 Experimental Environment

Figure 6.1 illustrates the experimental testbed that we used to collect traces for classification. There are four primary components to the setup: wireline nodes, wireless nodes, one commodity access point (AP), and a monitor node. The wireline and wireless nodes are connected in a dumbbell topology via the APs and a switch. The AP is a Cisco AP1200, running IOS version 12.3(8), with single Rubber Duck antenna, and integrated 802.11a module/antenna. The switch is a commodity LinkSys 10/100 16-port Workgroup Hub. The wireline nodes and switch are connected to the AP via 100Mbps Ethernet connections. The maximum data rate for 802.11a is 54Mbps. Having 100Mbps wireline links insures that the wireless, rather than the wireline, part of the network is the throughput bottleneck.

The wireline and wireless nodes are identically configured Sun 4200 AMD Opteron 275 (dual Core) nodes, with 4 GB RAM, Intel 82546EB (e1000) chips, running CentOS 5.2. The wireless nodes are installed with R52-350 mini-PCI cards (Atheros 5414 chip). We used default wireless interface configurations for our experiments. These were: *(a)* no RTS/CTS, and *(b)* no MAC layer fragmentation.

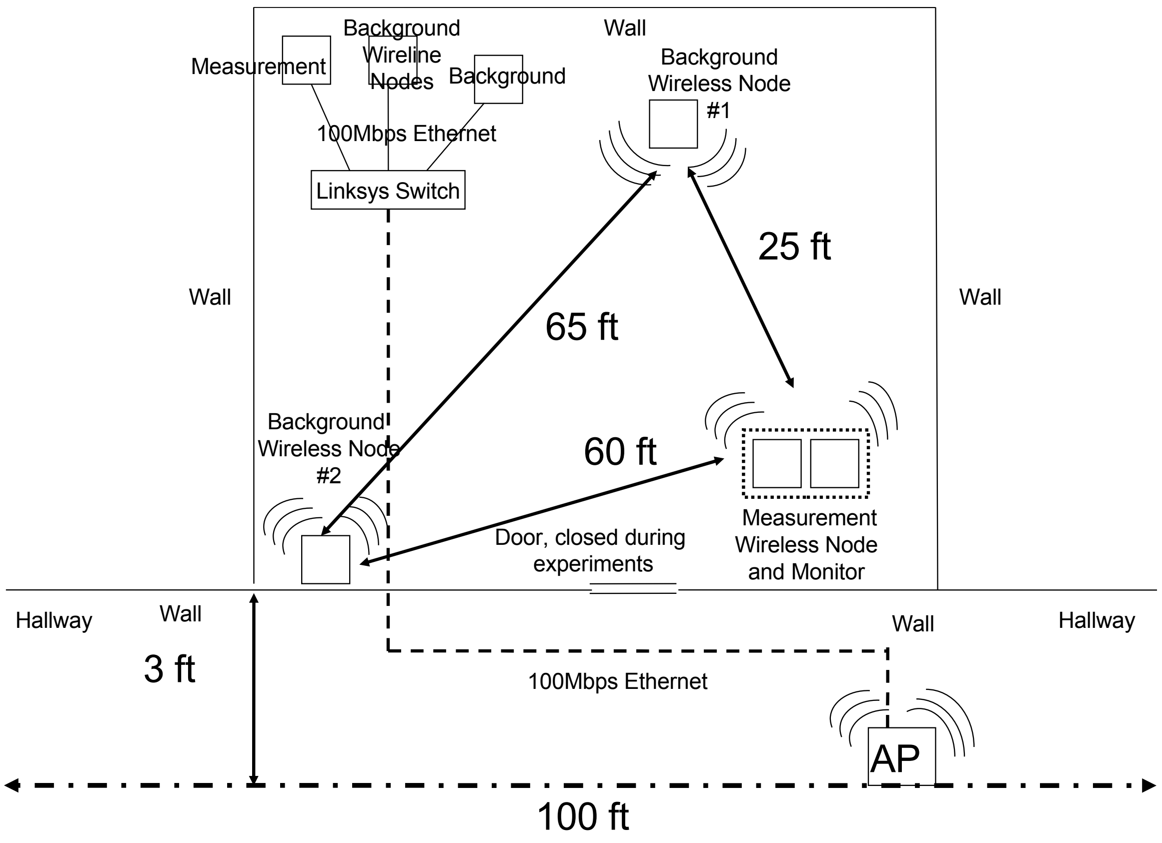


Figure 6.1: Laboratory testbed used to generate wireless network traces for rate adaptation algorithm classification.

Communication between wireless and wireline nodes is pairwise, *i.e.*, during an experiment, each wireless node sends data to a single, pre-assigned, wireline node and vice versa. One wireline-wireless pair is designated the measurement pair. Algorithm classification is done for packet traces from the measurement node pair. The other two node pairs generate background traffic. We refer to them as background pair one and background pair two. The monitor, an *AirPcapNx* adapter [1] is located next to the measurement node for all experiments. We use 802.11a channel 36 for all our experiments.

6.3.2 Experimental Protocol

The measurement node transferred 8MB files with 5 seconds between transfers. There were three levels of background traffic: no background traffic, one node pair generating background traffic, and two node pairs generating background traffic. The first background node pair transfers 4MB files with an interval of 2 seconds between transfers, and the second pair transfers 512KB files with 1 second between transfers. Both background nodes use *Sample Rate*, the default MadWifi rate adaptation algorithm.

For each background traffic level, 100 files per rate algorithm were transferred for each of the four algorithms. The file transfers for the different algorithms were interleaved in sets of 25 to compensate for possible external interference that could effect the integrity of classification.

We process packet traces for rate adaptation algorithm classification in the following manner. We generate a training/test feature vector for every instance where the transmission rate of the *kth* and *k+1th* 802.11 data packet transmitted by the measurement node is different. The rate transition is the center of the packet window over which we compute features. The feature values were normalized by subtracting the mean of each feature from the respective feature values and then dividing by the standard deviation before training and testing.

The training set consists of 10% of feature vectors selected uniformly at random from the first half of the transfers at each background traffic level. The test set consists of all feature vectors from the second half of transfers at each background traffic level. We constructed 5 different training sets via random sampling, and tested all of them using the second half of the transfers. The classification accuracy values in Section 6.4 are the averages of the five runs.

In Section 6.4, we present algorithm classification accuracy in terms of two metrics, *transfer classification accuracy* and *sample classification accuracy*. The primary accuracy metric is *transfer classification accuracy*. After all the feature vectors from an 8MB transfer in the test set have been classified, we determine the algorithm class of the majority of the feature vectors and designate that as the algorithm class for the whole transfer. The *transfer classification accuracy* values presented in Figures 6.2–6.5 refer to the percentage of 8MB transfers in the test set that are classified accurately, averaged over the 5 different training and test runs. We chose a metric that reports classification accuracy at the granularity of whole transfers rather than individual feature vectors for two reasons. First, the use of the majority vote based on a larger number of feature vectors increases robustness compared to classification based on a single feature vector. Second, in practice, algorithm classification is likely to be done for a complete user session or transfer, so our use of the majority vote to boost robustness does not require us to make any unrealistic assumptions. However, consideration of classification accuracy at the granularity of individual feature vectors cannot be ignored entirely for the following reason. A transfer classification is considered correct whether 100% of feature vectors from the transfer are classified correctly, or (assuming 4 algorithms) whether 25.1% of the feature vectors are classified correctly and the incorrect classifications split evenly between the remaining 74.9% feature vectors. However, the confidence in the classification is clearly much more in the first case. To guard against this accuracy confidence inflation, we also report individual feature vector classification accuracy. The *sample classification accuracy*[1] metric in Figures 6.6–6.9 refers to the fraction of individual feature vectors in the test set that are classified accurately, averaged over the 5 different training and test runs.

6.4 Building a Robust Classifier

In this section, we first describe the input features we used to construct the SVM-based classifier for algorithm identification. We then present results to demonstrate the high accuracy of our classifier, and discuss how feature selection affects accuracy. We conclude the section by providing evidence that the classifiers are portable, *i.e.*, that a classifier trained in one RF environment can successfully identify algorithms in a different RF environment.

6.4.1 Feature Selection for Algorithm Classification

For each rate transition event in the packet trace, *i.e.*, each time the transmission rate of the *kth* and *k+1th* packet differs, we compute two feature sets, detailed below, over windows of varying sizes. The difference in the feature sets is that they represent information at varying levels of granularity for training and testing. *Feature Set 1* aggregates the information in the feature vector window, while *Feature Set 2* exposes detailed packet trace information. The results in Section 6.4 will show that having packet trace information at varying levels of granularity is essential for accurate classification.

Feature Set 1: The following features are computed for a window of size m_1 around a rate transition event. Features *(1)–(4)* are computed once for the entire window, and *(5)–(12)* are computed individually for each of the eight 802.11a data rates. Each feature in the list below corresponds to two distinct features, one computed for a window of size m_1 before the rate transition event, and the second for a window of size m_1 after the rate transition event.

(1) The number of packets in the window.

(2) The packet reception probability, defined as the number of non-retry packets divided by the total number of packets in the window.

[1] In the remainder of this chapter we use the terms *individual feature vector* and *sample* interchangeably, based on ease of exposition

(3) The fraction of packets in the window that are unique, defined as the number of distinct 802.11 sequence numbers observed divided by the total number of packets.

(4) The trace accuracy, which is defined as $\frac{1-\# \ of \ missing \ 802.11 \ sequence \ numbers}{total \ \# \ of \ packets}$.

(5) The number of packets with each rate in the window.

(6) The number of packets with each rate that are retries in the window.

(7,8,9) The minimum, median, and maximum distance in packets for packets with each rate from the center of the window. Distance in packets is calculated in terms of the number of packets in the trace rather than packet sequence numbers. When a packet of a particular rate does not occur in a window, distance is set to a constant high value.

(10, 11, 12) The minimum, median, and maximum distance in packets for a retry packet with each rate from the center of the window.

Hence, for any value of m_1, there are a total of 136 features, $2 * 4$ for the four features (*1–4*) before and after the center of the window, and $8 * 8 * 2$ for the eight features (*5–12*) for each of the eight rates before and after the center of the window.

We consider the following different values of m_1: 100ms, 200ms, 300ms, 400ms, 500ms, 10 packets, 20 packets, 30 packets, 40 packets, 50 packets, 1-5 802.11 retries around a packet transition event.

Feature Set 2: For this set, the following three features corresponding to each packet in a window of size m_2 are included in the feature vector:

(1) The transmission rate of the packet.

(2) A binary value indicating whether the packet is a retry.

(3) The difference in sequence numbers between the packet and the center of the window.

We use the following values of m_2: 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50 packets. In this case, since there are 3 features per packet and the window size is m_2 packets before and after a rate transition, the total number of features per feature vector is $3 * ((2 * m_2) + 1)^2$.

Finally, we constructed an *all features* vector, which is the concatenation of feature vectors for all window sizes for both feature sets, and contains 3720 features. In Section 6.4, we present classification accuracy results for *all features*, all window sizes for *Set 2*, and 10, 20, 30, 40 and 50 packet window sizes for *Set 1*, because these feature sets yielded the most interesting results.

We use the multi-label SVM software SVM^{multiclass} [4], with a linear kernel and default parameters, for algorithm classification.

6.4.2 Evaluation of Classification Accuracy

We conducted two sets of experiments one week apart, which we refer to as *Experiment 1* and *Experiment 2*, both using the protocol described in Section 6.3.2. For results presented in this section, training and test sets were drawn

^{2}+1 for the packet at the center of the window.

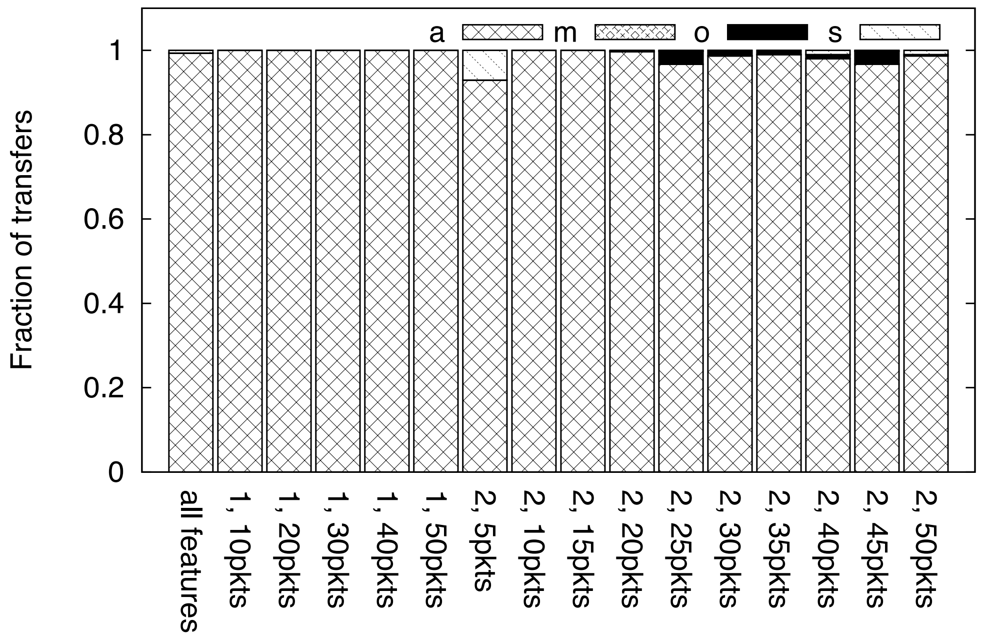
from the same experiment. For results presented in the next section, the classifier trained on data from *Experiment 1* was tested on data from *Experiment 2* and vice versa to investigate the portability and robustness of the classifier.

Figures 6.2–6.5 present our results in terms of the *transfer classification accuracy* metric. The stacked histograms illustrate the fraction of transfers classified for each algorithm for *Experiment 1* and *Experiment 2*. In each case, the training and test set were drawn from the same experiment. The figure labels indicate the experiment number and the correct algorithm. The x-axis indicates the feature set (1 or 2) and window size used. The y-axis indicates the fraction of transfers classified for a certain algorithm and feature set, *e.g.*, in Figure 6.4b, *Set 1, 20 pkts* for *Experiment 2, Onoe* shows that approximately 70% of transfers were classified correctly as *Onoe*, 25% were classified incorrectly as *Minstrel* and 5% were classified incorrectly as *AMRR*. Algorithms are indicated in the key with the first letter in their name.

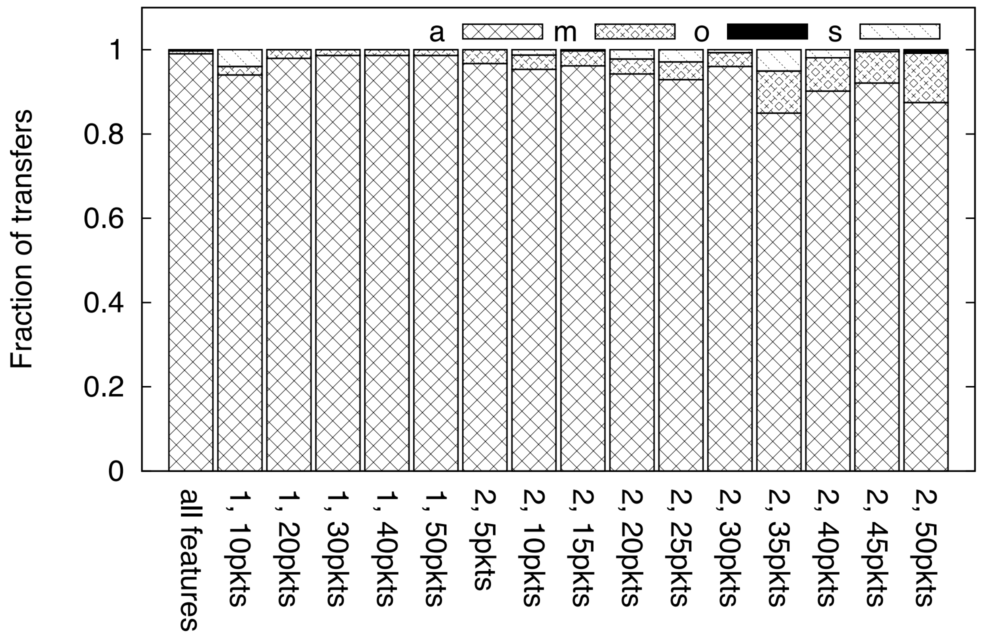
We present results from two different runs of the same experiment, conducted in the same laboratory environment but one week apart, because wireless network conditions are difficult to replicate due to external interference effects. Table 6.1 presents the distribution of transmission rates for the two experiments, and it can be seen that there is a significant difference in the distributions in many cases.

Figures 6.2–6.5 show that for all algorithms except *Onoe* (Figure 6.4) the classification accuracy for *all features* is high, ranging from 95% to 100%. Also, for all cases except *Experiment 1, Onoe* (Figure 6.4a), the accuracy for *all features* is very close to that of the highest accuracy individual feature set for each experiment. Both observations support our learning-based approach. Their combination means two things. First, classification accuracy is high. Second, feature set and window size selection are simplified because the concatenation of all feature sets and window sizes in *all features* results in accuracy equal to that of the most accurate individual feature set. This occurs even though the most accurate feature set varies by algorithm and experimental run.

The classification accuracy for *Onoe* is lower than all other algorithms across all feature sets. This is explained by considering the *Rate Change* column of Table 6.1. This column indicates the rate change frequency, *i.e.*, the percentage of times two consecutive packets in a trace have different transmission rates. *Onoe* has the lowest rate change frequency, 4.1% to 7.7%. *Sample Rate* has the next lowest rate change frequency, 9.7% to 13.2%, roughly twice that of *Onoe*. This difference means that over a given packet window, a *Sample Rate* trace will provide SVM with twice as much information to generate a distinguishing signature compared to *Onoe*, resulting in higher classification accuracy for *Sample Rate*. The fact that *Onoe* has by far the lowest rate change frequency is consistent with what we know about the four algorithms. *Onoe* is the slowest to change rates because of its use of credits to calculate rate change. Also, it has the longest and slowest-to-change retry chain with a *c0* value of 4, which further reduces the frequency of rate change and hence the information content of packet traces. All other algorithms change rates at a frequency of roughly 11%-30% due to shorter retry chains and properties such as sampling alternate rates or trying a higher rate after every 10 successful packet transmissions at a given rate, yielding traces with a higher information content and therefore better classification accuracy.

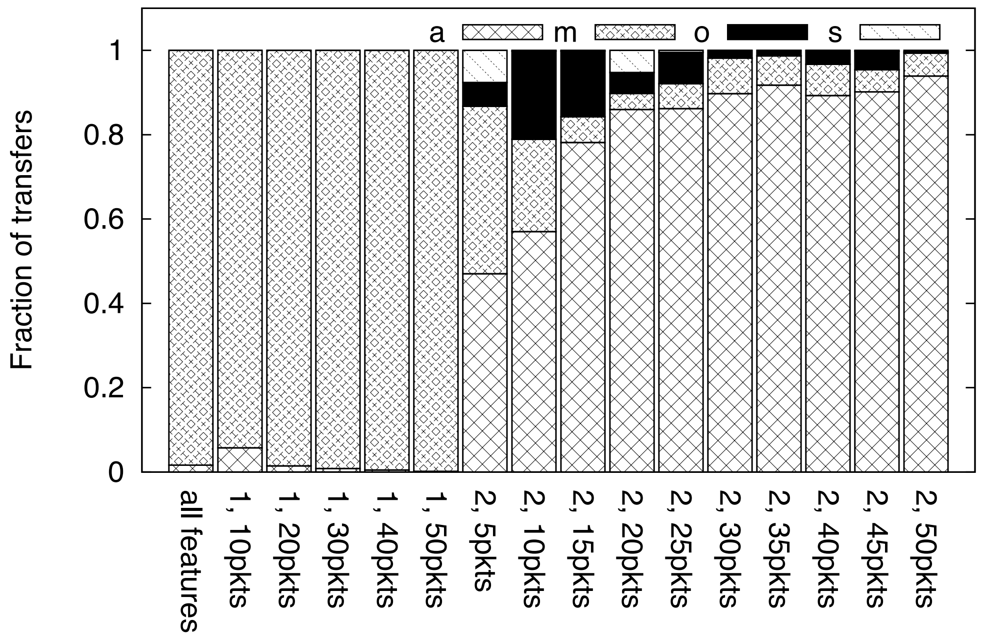


(a) Experiment 1

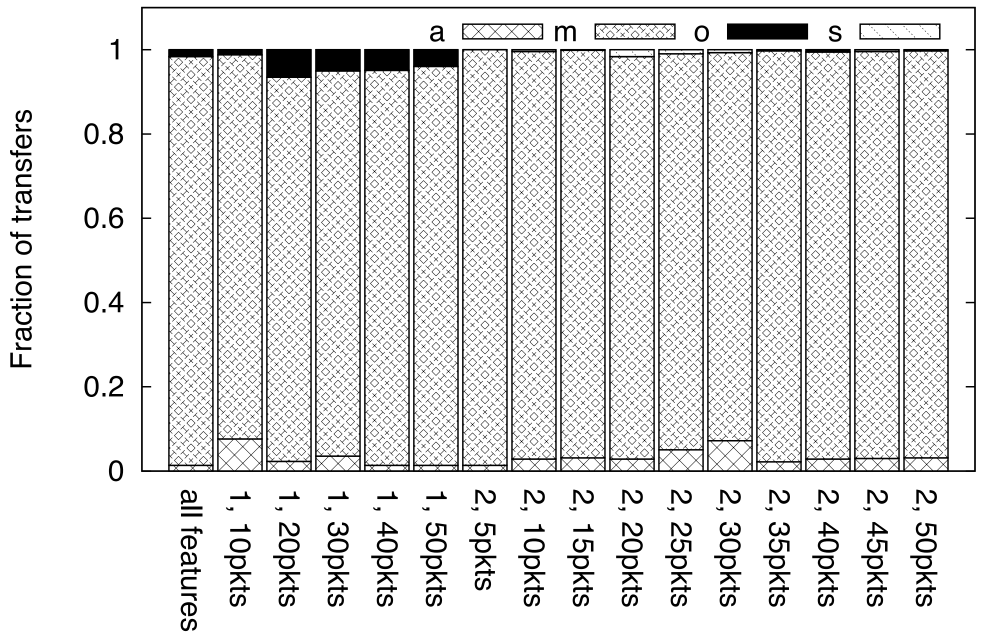


(b) Experiment 2

Figure 6.2: Transfer classification accuracy for *AMRR*

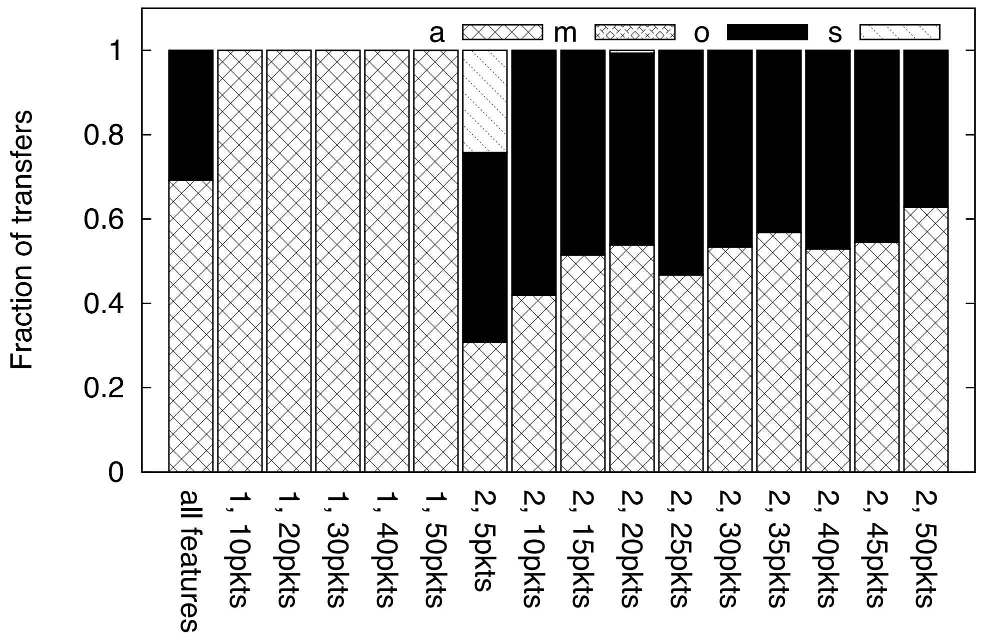


(a) Experiment 1

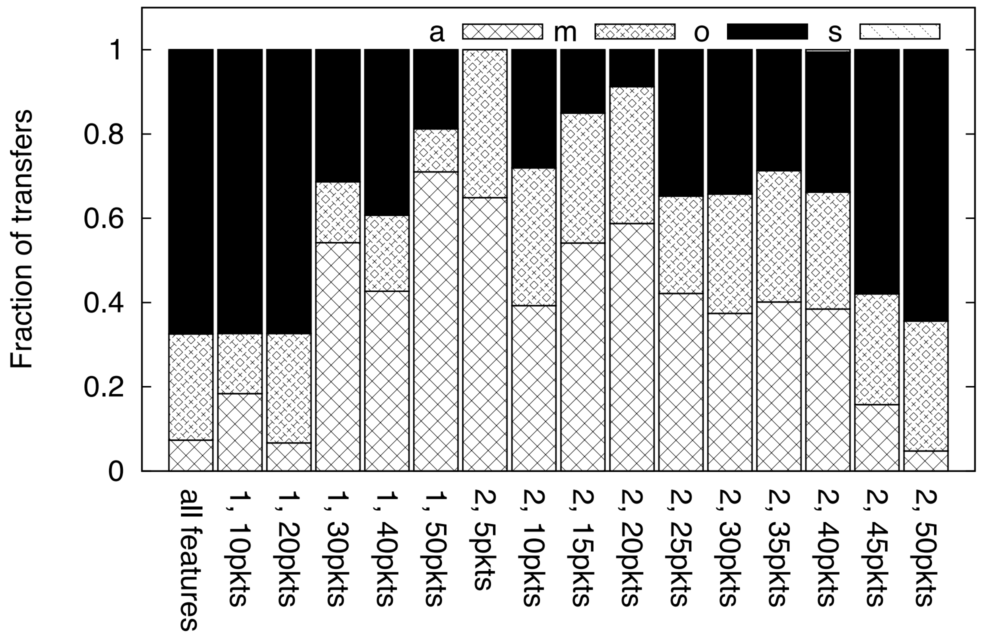


(b) Experiment 2

Figure 6.3: Transfer classification accuracy for *Minstrel*

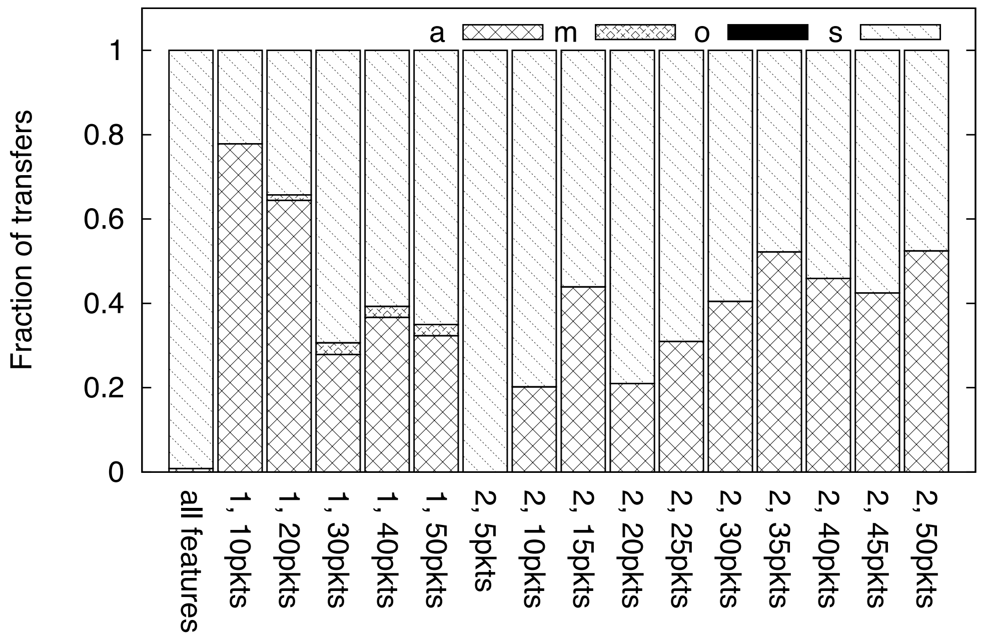


(a) Experiment 1

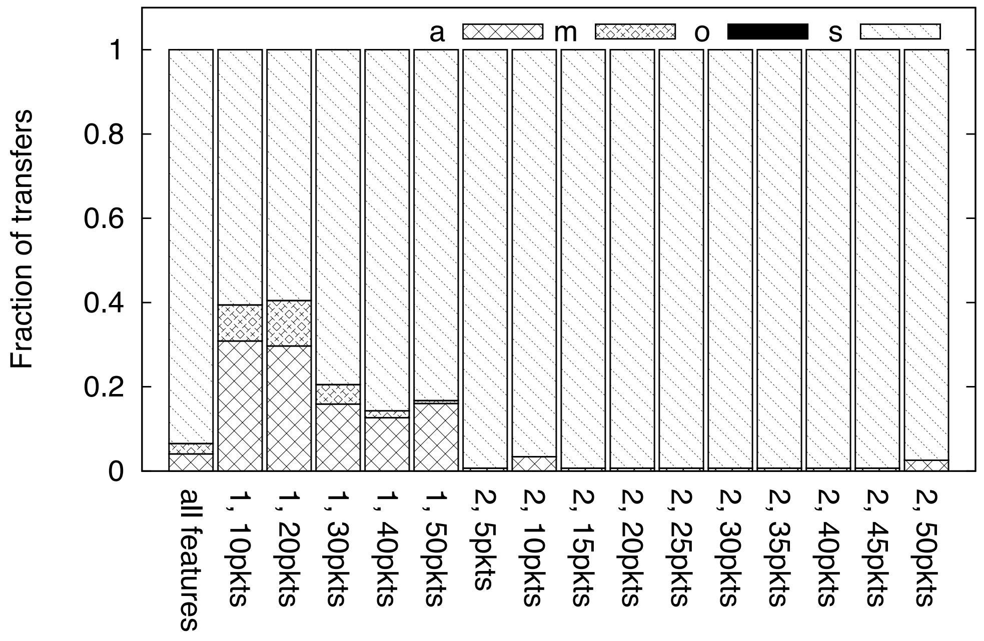


(b) Experiment 2

Figure 6.4: Transfer classification accuracy for *Onoe*

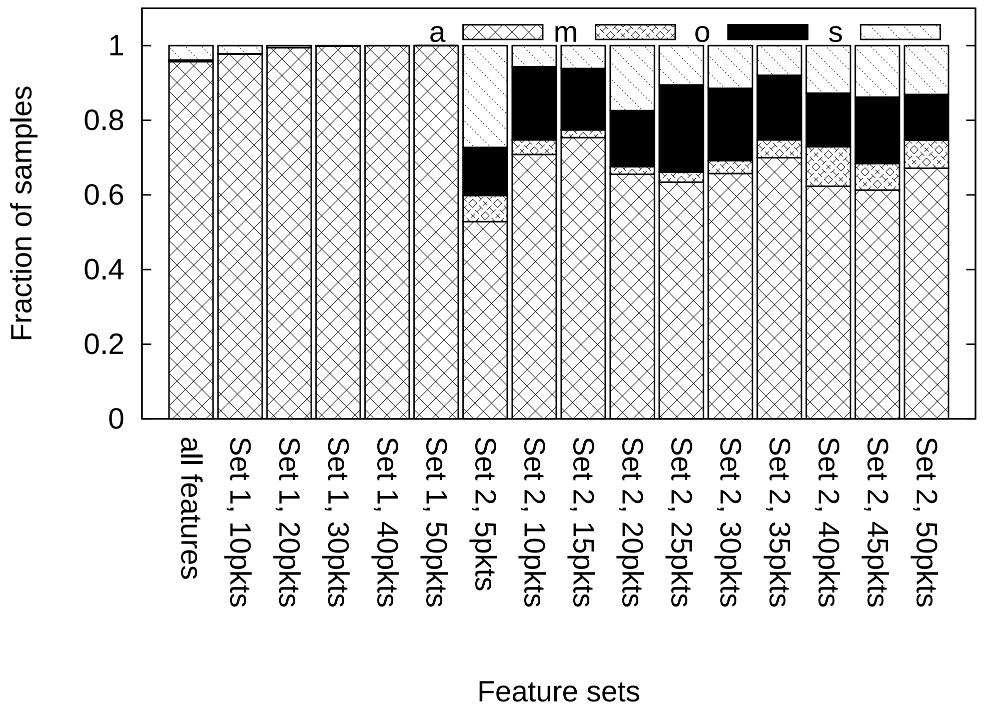


(a) Experiment 1

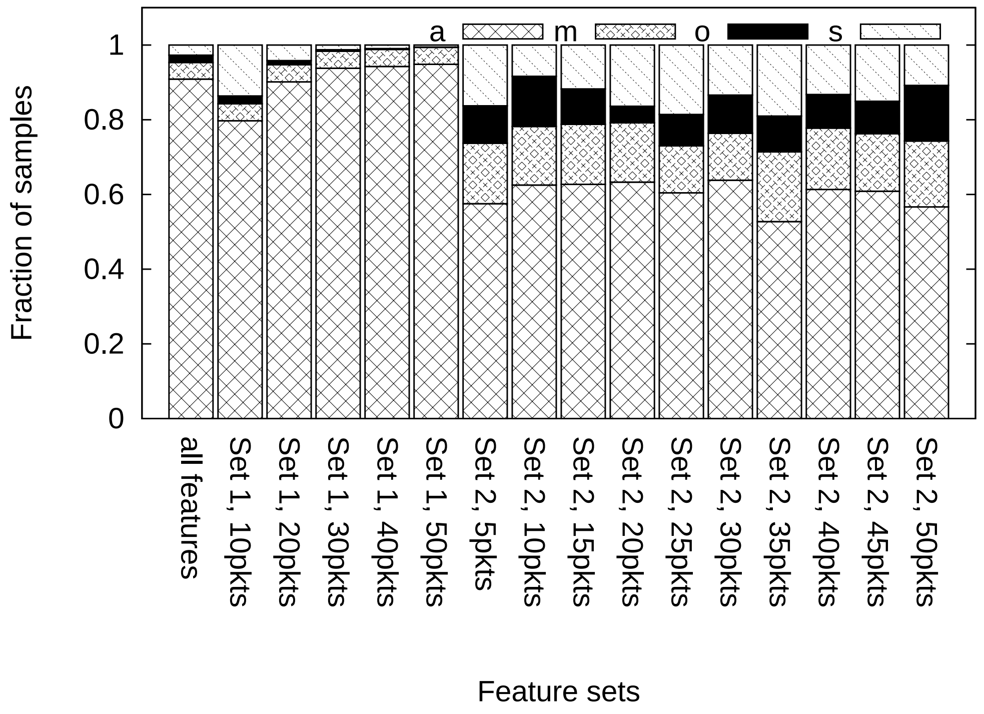


(b) Experiment 2

Figure 6.5: Transfer classification accuracy for *Sample Rate*

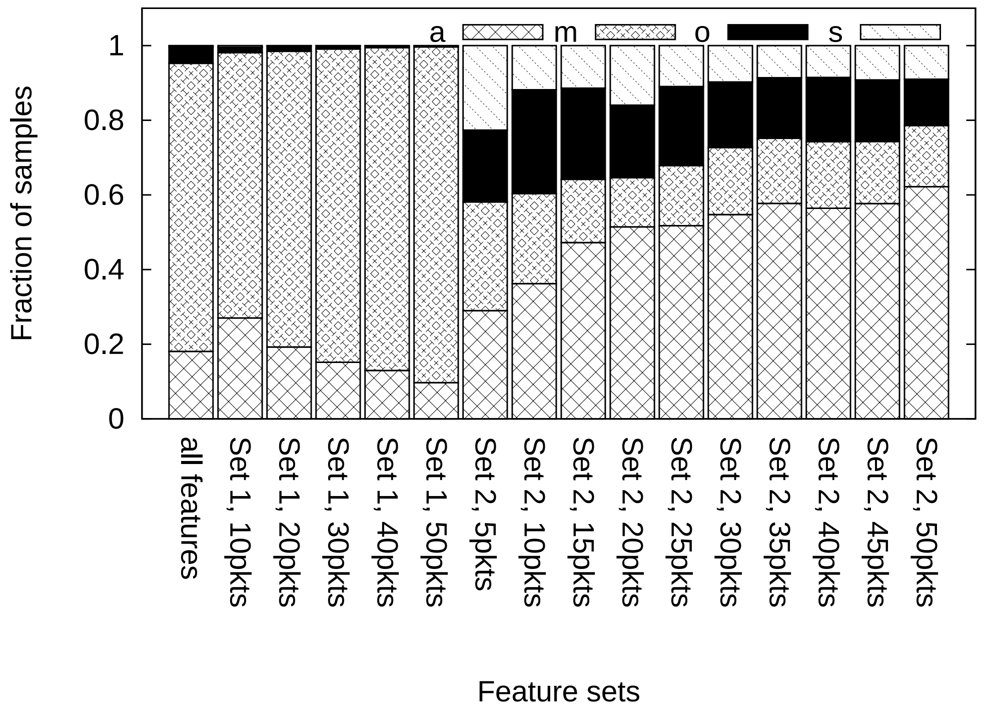


(a) Experiment 1

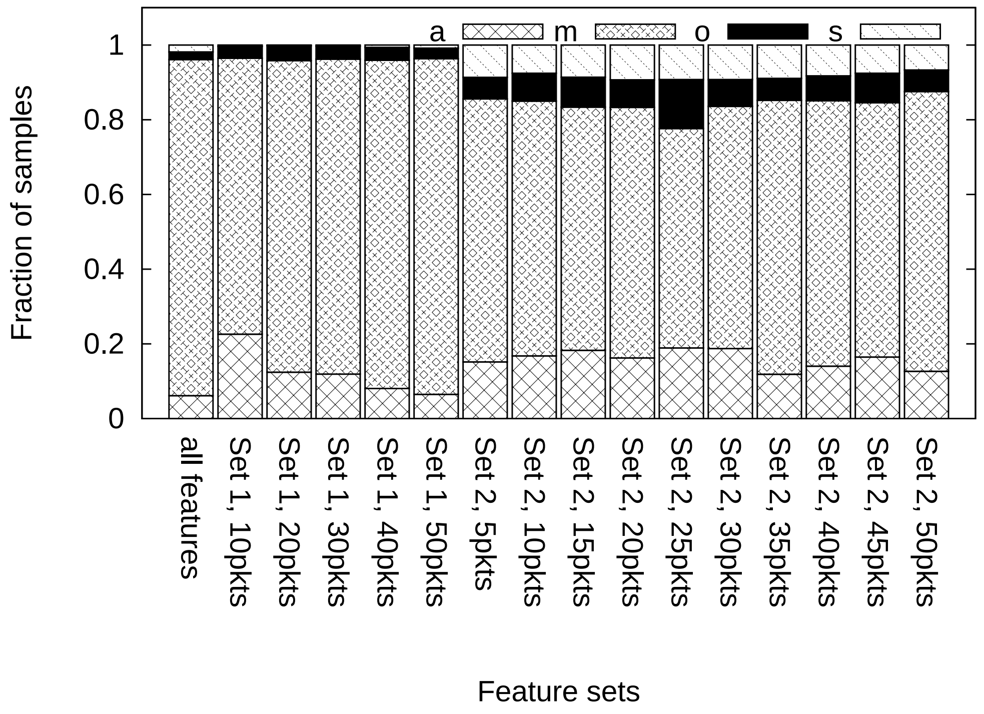


(b) Experiment 2

Figure 6.6: Sample classification accuracy for *AMRR*

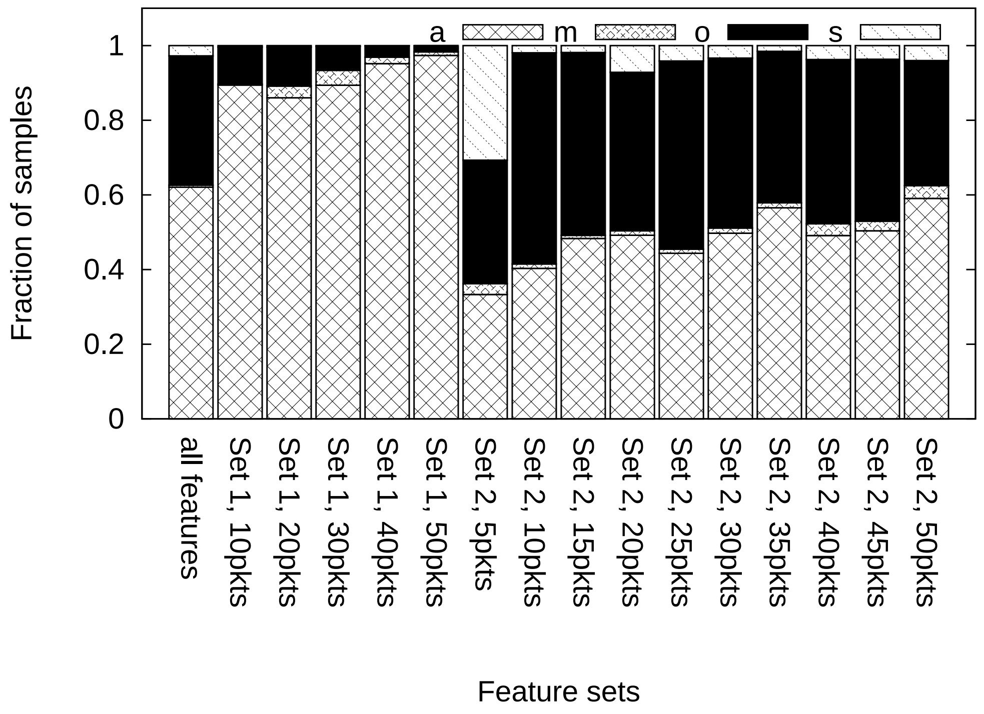


(a) Experiment 1

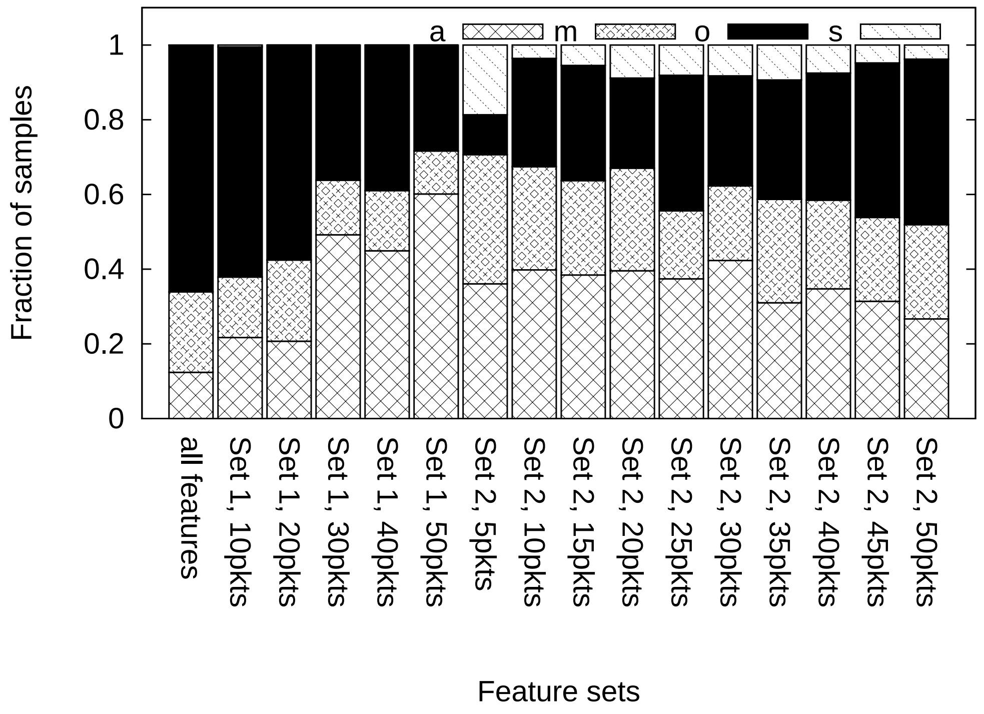


(b) Experiment 2

Figure 6.7: Sample classification accuracy for *Minstrel*

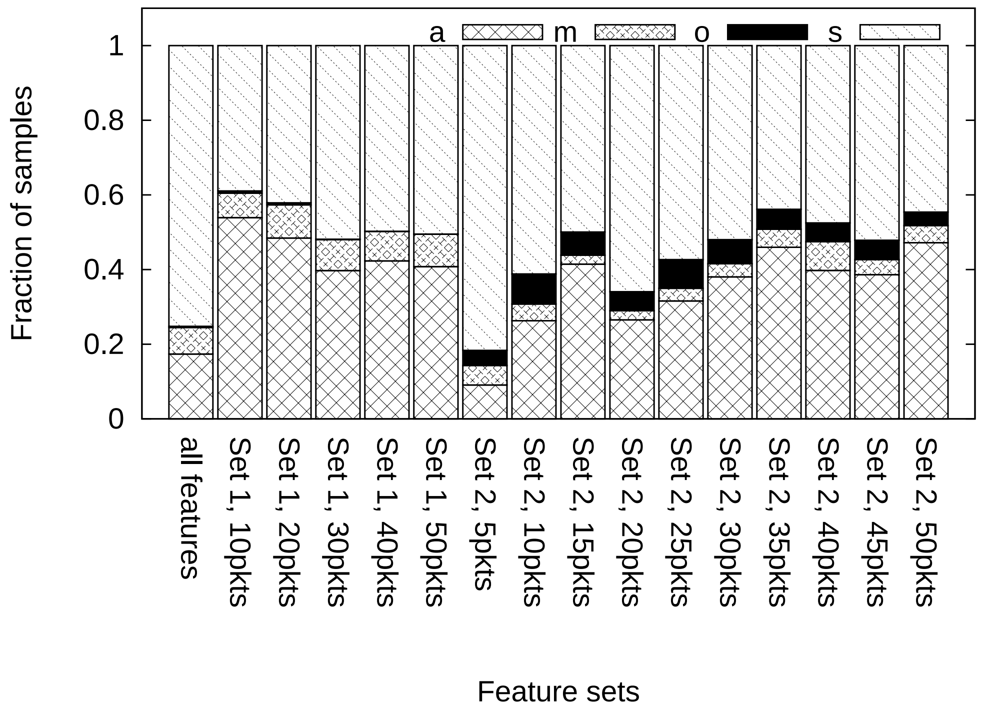


(a) Experiment 1

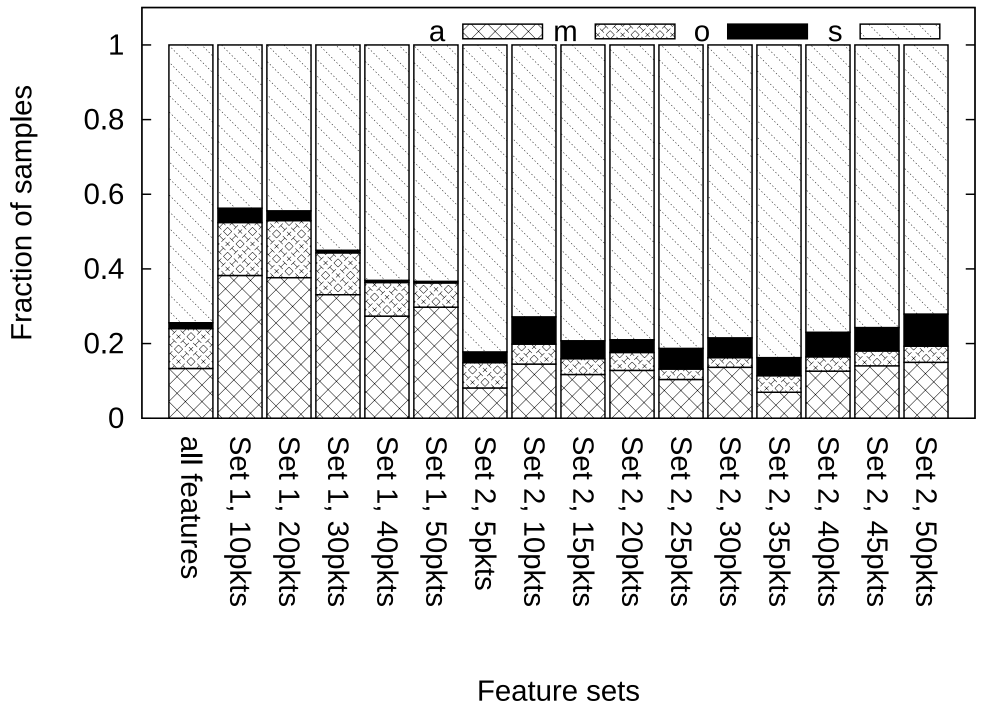


(b) Experiment 2

Figure 6.8: Sample classification accuracy for *Onoe*



(a) Experiment 1



(b) Experiment 2

Figure 6.9: Sample classification accuracy for *Sample Rate*

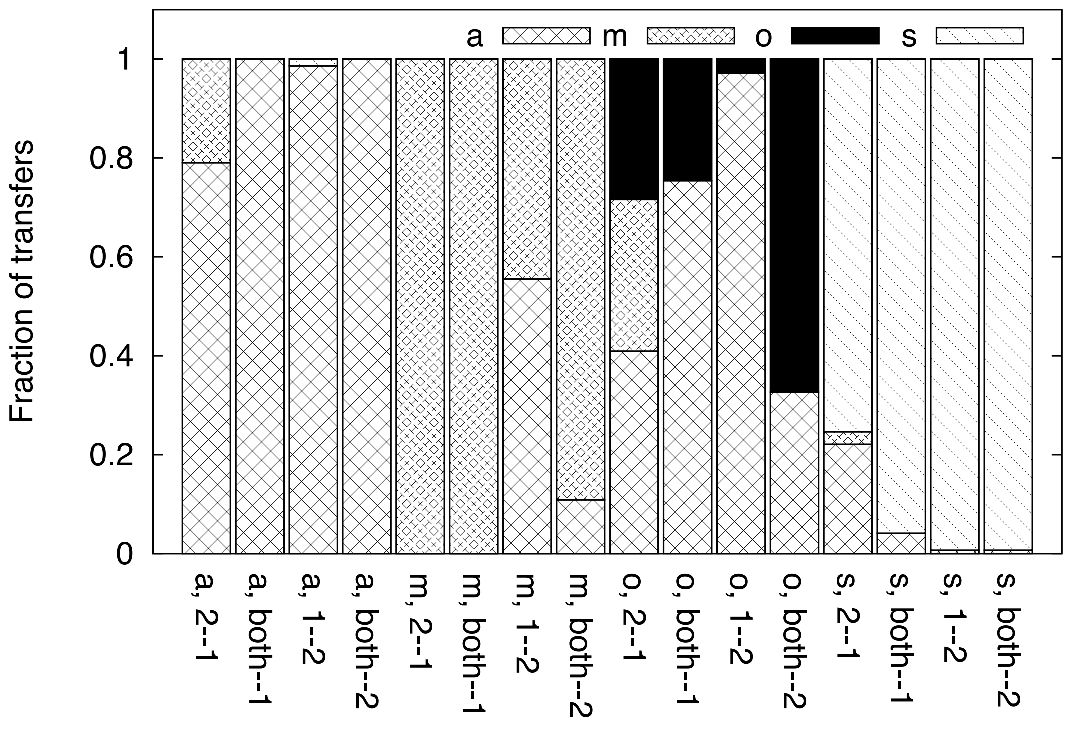


Figure 6.10: Stacked histogram showing the fraction of transfers classified correctly and incorrectly for each algorithm, training set and test set combination. This figure illustrates the potential for the portability of the classifier. All results presented are for the *all features* set. The x-axis indicates the correct algorithm and the training and test sets, *e.g., a, 2–1* indicates that AMRR is the correct algorithm, the training set consisted of data from *Experiment 2* and the test set consisted of data from *Experiment 1*, while *m, both–2* indicates that Minstrel is the correct algorithm, the training set consisted of data from both *Experiments 1 and 2*, and the test set consisted of data from *Experiment 2*.

Table 6.1: Distribution of packets across transmission rates for four target algorithms. The first column indicates the algorithm (algorithms are identified by the first letter in their names), the background traffic level (nobg for no background traffic, bg1 for one background node pair generating traffic, and bg2 for two background node pairs generating traffic), and the experiment number (1 or 2). The columns labeled with rates indicate the percentage of packets in each experiment transmitted at that rate. *Rate Change* indicates the percentage of times two consecutive packets were transmitted at different rates. *Retry* indicates the percentage of packets that were retries. *Consecutive Retry* indicates the percentage of times the *k+1th* packet was a retry given that the *kth* packet was a retry. This is an estimate of how far down the retry chain the algorithm had to go in case of a loss. The last three values are measures of the information content of a packet trace. Algorithms that have higher values are likely to be classified with higher accuracy because they provide SVM with a larger amount of information over a given time window, enabling SVM to generate a better distinguishing signature. All values are aggregates for all transfers in a given experiment.

Experiment	**6Mbps**	**9Mbps**	**12Mbps**	**18Mbps**	**24Mbps**	**36Mbps**	**48Mbps**	**54Mbps**	**Rate Change**	**Retry**	**Consecutive Retry**
a, nobg, 1	0.3	0.0	0.1	0.7	9.6	78.1	11.1	0.0	31.8	16.5	6.2
a, nobg, 2	0.4	0.1	0.6	1.3	15.3	77.2	5.0	0.0	30.3	15.7	6.2
a, bg1, 1	0.3	0.1	1.3	6.7	37.7	51.2	2.8	0.0	28.6	15.4	9.8
a, bg1, 2	0.6	0.6	2.8	8.7	35.7	50.0	1.6	0.0	29.1	15.7	10.9
a, bg2, 1	0.3	0.7	1.4	5.4	31.3	55.4	5.5	0.0	30.7	16.4	9.2
a, bg2, 2	0.6	0.5	2.4	8.1	48.7	39.4	0.3	0.0	31.6	17.0	11.0
m, nobg, 1	0.0	0.0	0.1	0.2	0.5	92.4	4.0	2.8	7.4	14.2	42.9
m, nobg, 2	0.3	0.0	0.0	0.6	2.8	89.7	3.9	2.5	10.3	23.0	46.8
m, bg1, 1	0.2	0.1	0.1	0.3	2.8	88.9	5.1	2.6	13.1	24.6	44.9
m, bg1, 2	0.5	0.1	0.2	1.5	15.5	75.9	3.9	2.4	13.4	27.8	47.8
m, bg2, 1	0.4	0.0	0.1	0.5	1.9	89.5	4.9	2.6	10.5	22.7	41.3
m, bg2, 2	0.6	0.1	0.3	1.2	12.6	78.5	4.3	2.4	14.9	30.3	50.8
o, nobg, 1	0.0	0.0	0.0	0.0	4.9	94.8	0.2	0.0	4.1	10.9	32.8
o, nobg, 2	0.2	0.0	0.0	0.5	46.7	52.6	0.0	0.0	5.8	15.1	38.7
o, bg1, 1	0.2	0.0	0.0	0.7	33.0	66.2	0.0	0.0	7.7	18.8	30.4
o, bg1, 2	0.3	0.0	0.1	1.8	81.8	16.0	0.0	0.0	5.4	17.3	26.8
o, bg2, 1	0.3	0.0	0.0	1.4	56.1	42.1	0.0	0.0	5.8	17.2	26.6
o, bg2, 2	0.3	0.0	0.1	1.9	82.4	15.2	0.0	0.0	5.1	17.2	26.7
s, nobg, 1	0.2	0.0	0.2	2.9	41.5	54.5	0.7	0.1	9.7	8.2	23.9
s, nobg, 2	0.4	0.0	0.5	11.9	61.7	25.2	0.2	0.0	10.7	9.0	25.9
s, bg1, 1	1.1	0.0	2.9	18.9	49.2	27.4	0.4	0.1	11.6	16.8	25.5
s, bg1, 2	0.8	0.0	4.5	24.7	57.2	12.7	0.1	0.0	11.3	16.3	25.3
s, bg2, 1	1.7	0.0	4.6	17.1	40.3	36.0	0.2	0.1	13.2	17.9	29.7
s, bg2, 2	0.9	0.0	4.7	22.5	54.6	17.0	0.1	0.0	11.8	16.6	25.8

Another observation from Figures 6.2–6.5 is that different window sizes and feature representations (*Set 1* versus *Set 2*) have different classification accuracies for the different algorithms and even for different experiments with the same algorithm. For example, *Set 1* has high classification accuracy for both experiments with *Minstrel*, while *Set 2* has high accuracy only for *Experiment 2* (Figure 6.3). Also, for *Sample Rate*, *[Set 2, 5 pkts]* is the only feature set and window combination that has high accuracy for *Experiment 1* (other than *all features*), while a number of feature set and window combinations have high accuracy for *Experiment 2* (Figure 6.5).

To explain why different window sizes and feature representations result in different classification accuracy, we have to consider how transmission rate changes between two successive packets. One reason for rate change is retry chain traversal due to packet loss. The second is sampling of other candidate rates, as in *Sample Rate* and *Minstrel*. The third is due to the algorithm deciding that the current *r0* is suboptimal and picking a new *r0*. Rate change due to the traversal of the retry chain occurs at the finest time granularity, from one packet to the next, for the length of the retry chain. Rate change due to sampling occurs at the second finest granularity (10 packets). Rate change due to computation of a new optimal *r0* occurs at the coarsest granularity (tens of packets, 1 or more seconds).

Consider a hypothetical algorithm that changes rates only due to retry chain traversal. Such an algorithm's distinguishing signature would be captured most effectively by highly detailed information over a small window, such as *[Set 2, 5 pkts]*, because all rate changes would occur at the granularity of individual packets. If *Set 1* and a larger window are used, the fine-grained information that is the algorithm's signature will be lost. Consider another hypothetical algorithm that only changes rate based on long-term loss and throughput feedback, *i.e.*, when it re-evaluates the optimal rate, it sets all rates in the retry chain to that one optimal value. Such an algorithm's distinguishing signature would be captured most effectively by features that *(a)* are computed over longer windows, and *(b)* can filter out short term variations and capture longer term throughput, loss and rate change behavior, such as *[Set 1, 50 pkts]*.

Practical algorithms change rates at all three time granularities, so features computed at varying levels of detail at a range of time granularities are necessary to comprehensively capture an algorithm's signature. This is why *all features*, a concatenation of all feature set and window size combinations, always has the highest classification accuracy. However, in a given run for an algorithm, rate might change for one particular reason more often than for others. If the dominant cause of change is retry chain traversal, we expect *[Set 2, 5 pkts]* to be the most accurate classifier, as for *Sample Rate, Experiment 1*. If, however, the dominant cause of change is reevaluation of the optimal rate, we expect *Set 1* classifiers to be more accurate than *Set 2* classifiers, as for *Minstrel, Experiment 1* (Figure 6.3a). Based only on passive packet traces, it is difficult to identify with certainity the cause of rate change from one packet to the next.

Next, we consider the *sample classification accuracy* results to ensure that there is no serious accuracy confidence inflation (Section 6.3.2) when the majority vote of sample classification is used for transfer classification. These results are presented in Figures 6.6–6.9. Each stacked histogram in Figures 6.6–6.9 corresponds to the histogram with the same label in Figures 6.2–6.5. We compare the transfer and sample accuracy for *all features*, the feature set with the highest transfer classification accuracy, for the cases where the transfer classification is correct, *i.e.*, for *AMRR*, *Minstrel* and *Sample* for both *Experiment 1* and *Experiment 2*, and *Onoe* for *Experiment 2*. We find that in all cases while sample accuracy is slightly less than transfer accuracy in the case of correct classification (*e.g.*, for *Sample, Experiment 1* the sample accuracy is 75% while the transfer accuracy is 99%), there are no major accuracy distortions due to considering accuracy at the granularity of transfers.

6.4.3 Evaluation of Classifier Portability

We investigate the portability of classifiers by testing the classifier trained on *Experiment 1* on transfers from *Experiment 2* and vice versa. We conclude that classifier portability depends on the relative information content in the training and test data. If the training data's information content is greater than or equal to that of the test data, the classifier is as accurate for the test data from the other experimental run as it is for test data from its own experimental run, otherwise it is less accurate.

Figure 6.10 illustrates the classification accuracy when a classifier trained on data from one experiment is tested on data from the second experiment. The classification accuracy for *Onoe* is low across all training and test sets due to low information content of traces, as discussed above. The classification accuracy for *AMRR* and *Sample Rate* is high (70% or higher) in all cases. Table 6.1 shows that this is the case even though the distribution of packets by transmission rate is different for the two experimental runs for each algorithm. What is similar about the two runs for each algorithm is the information content of the traces measured in terms of the last three metrics in Table 6.1.

The interesting case in Figure 6.10 is *Minstrel*, where the classifier trained on data from *Experiment 1* performs poorly on test data from *Experiment 2* (case *m, 1–2*) with an accuracy of only 40%, but the classifier trained on *Experiment 2* performs very well on *Experiment 1* (case *m, 2–1*) with an accuracy of 100%. Table 6.1 shows that for all the three measures reported, *Set 2* has higher information content than *Set 1*. This means that the classifier trained on *Experiment 1* captures a limited set of the algorithm behavior compared to that present in *Experiment 2* test data, resulting in poor classifier accuracy. However, when the classifier is constructed with data from both *Experiment 1* and *Experiment 2*, its accuracy on *Experiment 2* tests increases dramatically, suggesting that enhancing a ported classifier with current data is beneficial.

These results have important implications for the practical use of our methods. They show that if the training set has high enough information content, the resulting classifier is portable, regardless of the difference in the distribution of packet transmission rates. Classifier portability is a desirable property in general because it reduces training time in new environments. Classifier portability is essential for wireless networks because the RF environment is constantly changing, and retraining every time is expensive. The results in this section also show that the accuracy of a poorly-performing classifier can be improved by updating it with training data from the current environment.

6.5 Practical Deployment

In this section, we discuss issues relating to the practical deployment of our 802.11 rate adaptation classification method.

We envision that a pre-computed classifier composed of detailed training sets – training sets that exercise the rate adaptation algorithms over a wide range of conditions and hence result in the most robust classifier – would be distributed for use by network analysts and administrators. The setup for trace collection can be exactly the same as the experimental setup from Section 6.3. The major practical consideration in trace collection is that the monitor should be located such that it captures a sufficient number of packets from the sender of interest – if too few packets are captured, classification accuracy will deteriorate. Characterization of the precise relationship between trace completeness and classification accuracy is a topic for future work.

Our study has considered only open-source rate adaptation algorithms. Wireless clients (and even APs) in production networks are unlikely restrict themselves to using only open-source 802.11 drivers, so it is important to identify closed-source proprietary algorithms as well. Proprietary algorithms can be classified based on their 802.11 driver or driver version. One potential challenge in trying to classify proprietary algorithms is that no algorithm knowledge is available to assist in the selection of feature vectors. However, our feature selection (Section 6.3), even though it is for open-source algorithms, contains features that are based on patterns of rate change visible through 802.11 packet headers rather than any information that requires explicit algorithm knowledge. Hence we believe that proprietary algorithms can be classified accurately based on our feature set, and the only potential changes that might be required would involve increasing the range of packet windows over which features are computed.

Chapter 7

Summary, Conclusions and Future Directions

In this dissertation, we proposed the use of Support Vector Machines (SVMs), a state-of-the-art machine learning technique, for problems in computer network performance analysis. SVMs can accurately and efficiently handle problems with large numbers of variables and complex non-linear relationships between variables. There is an increasingly urgent need for such a technique for computer network performance analysis as networks become more complex due to the deployment of increasingly sophisticated technologies and protocols and due to the increase in network size in terms of users, available bandwidth, and the volume of data transmitted.

To determine whether SVMs are indeed a suitable framework for solving network performance problems in the face of increasing network complexity, we examine their effectiveness in solving three network analysis problems, (a) TCP throughput prediction for wireline networks, (b) TCP throughput prediction for wireless networks, and (c) identification of the 802.11 rate adaptation algorithms deployed in a network. The difficulty of classification and prediction problems increases with network complexity, so these problems provide appropriate test cases for evaluating the SVM framework. The difficulty of classification problems increases with network complexity because the more sophisticated the algorithms and protocols, the harder it becomes to identify them using conventional methods such as explicit examination of short packet sequences for characteristic packet sizes, packet interarrival times, and other signature patterns. The difficulty of prediction problems increases with network complexity because the number of variables that effect the target metric increases, and the relationship between the variables and target metric becomes more complicated.

The effectiveness of network performance analysis methods is evaluated based on several criteria. The first and most important criterion is the accuracy of the method. The second criterion is *analysis comprehensiveness*, *i.e.*, whether the method can handle fully general production networks or whether there is a need for simplifying assumptions. The third is *extensibility and maintenance*, a measure of how easily the method can be updated to accomodate modified or optimized versions of the environments the method was originally designed for. The fourth is *robustness*, *i.e.*, whether the method can maintain high accuracy over a wide range of operating conditions. The fifth is *agility*, a measure of how quickly the method responds when there is a change in operating conditions. The sixth is *practicality*, which considers factors relating to the ease of deployment, such as measurement traffic overhead and the availability of system information required for the method to work. Ideally, any performance analysis method would meet all of the above-mentioned criteria. However, in practice it is not always possible to satisfy all the criteria, so effective analysis methods often have to make compromises. Our goal in developing SVM-based solutions to network analysis problems is to minimize such compromises.

In the remainder of this chapter, we recap the results of our investigations, focusing on how far the SVM-based solutions meet the above-mentioned criteria. We conclude the chapter with a discussion of directions for future work.

7.1 Wireline TCP Throughput Prediction

The first problem for which we devised an SVM-based solution is TCP throughput prediction for wireline paths. Throughput prediction is useful in scenarios where there are multiple paths between senders and receivers, such as is the case with overlay networks and multi-homing, because it facilitates selection of the best path, *i.e.*, the highest throughput path, for a file transfer.

We predict throughput by constructing an SVM-based mapping between TCP throughput and path properties such as packet loss, available bandwidth and queuing delay. The first step of our investigation was conducted in a laboratory environment that provided highly accurate passive measurements of network path properties. Using these high accuracy passive measurements, the SVM-based predictor predicts throughput within 10% of actual 87% of the time under heavy traffic conditions (90% average utilization on the bottleneck link), a nearly 3-fold improvement on the most accurate predictor in the prior work. Using active measurements of path properties that can be conducted in real wide area paths, 49% of SVM-based predictions were within 10% of actual, which, while lower than accuracy based on perfectly accurae passive measurements, is still a factor of 1.6 better than the best method in prior work. Hence the SVM-based predictor comfortably meets the high accuracy criterion. Prior methods only handle TCP throughput prediction for bulk transfers. We show that the SVM-based method can extrapolate with high accuracy from 3 file sizes in the training set to a wide range of file sizes in the test set.

We found that of the three path properties considered, packet loss and queuing delay were sufficient for this high accuracy, and that the inclusion of available bandwidth measurements did not improve accuracy any further. This finding is important because active measurements of available bandwidth are considerably more heavyweight than loss and queuing delay measurements. The measurement overhead of the SVM-based method using active measurements is a factor of 13 lower than that of the most accurate method in prior work, so the SVM-based method meets the practicality criterion.

Prior methods have shown that level shifts in network conditions pose a challenge for TCP throughput prediction, with predictors often being slow to respond. We show that the SVM-based method needs only one training sample of new path conditions to resume highly accurate predictions. Hence the SVM-based method meets the agility criterion.

We test the SVM-based method on real wide area paths. These paths include trans-Atlantic and trans-continental-U.S. paths with RTTs ranging from 8 ms to 145 ms. Our method demonstrates high accuracy on these paths. The fact that our method performs well over a wide range of path conditions is evidence of its robustness. The fact that it performs well on real wide area paths is evidence that it meets the analysis comprehensiveness criterion.

7.2 Wireless TCP Throughput Prediction

The second problem for which we devise an SVM-based solution is TCP throughput prediction for wireless networks. This is a very different and much more complicated problem compared to TCP throughput prediction of wireline networks because PHY and MAC layer dynamics and environmental factors such as interference and multi-path effects have a large impact on wireless throughput. Hence wireline TCP throughput prediction methods cannot be trivially adapted for wireless environments.

Our SVM-based approach is targeted towards *opportunistic wireless networks*, particularly *vehicular* opportunistic networks. Opportunistic networking is a connectivity model in which guest clients in the vicinity of open APs utilize the APs to obtain temporary network access. Vehicular opportunistic networking applications are being explored by many research efforts because the high density of APs in urban areas provides many network access opportunities to vehicular clients.

Existing wireless TCP throughput prediction approaches are unsuitable for use in the vehicular scenario for two reasons. First, these approaches require tens or hundreds of seconds to generate a prediction, and vehicular clients remain within range of a given AP for only about 10 seconds, requiring a prediction to be generated in about 1 second to be of use to applications. Second, these approaches require knowledge about, and cooperation of, all other wireless senders in the vicinity, when in the vehicular opportunistic scenario only communication between the target client and the open APs can be assumed. Hence there is a need for a new wireless TCP throughput prediction technique for the vehicular scenario that (a) is accurate, (b) meets the stringent prediction latency requirements, and (c) is based on measurements that can be carried out in practice.

Our SVM-based technique predicts wireless throughput using very short (0.3 – 1.25 second) active probes between the AP and the target vehicular client. It assumes no explicit knowledge or cooperation of any other elements in the network. We test our method in a wide variety of network conditions, including variable background traffic and using stationary, walking and driving nodes. We find that, using the 0.3 – 1.25 second probes, our method can predict throughput within a factor of 2 of actual 80% to 100% of the time for all our network scenarios. While this accuracy level is much lower than what we achieved for wireline TCP throughput prediction, this accuracy level is useful for some, if not all, applications, and the accuracy bound holds even when clients are moving at speeds of 15–25 mph. Hence our SVM-based method yields reasonable accuracy while meeting the stringent time requirements of vehicular clients and maintaining realistic assumptions, *i.e.*, only relying on communication between the target client and the AP.

7.3 802.11 Rate Adaptation Algorithm Fingerprinting

The third and final problem for which we devise an SVM-based solution is identification of the 802.11 rate adaptation algorithms deployed in a network. Rate adaptation algorithms have a large impact on wireless throughput. Many efforts have focused on developing and understanding the behavior of rate adaptation algorithms in test environments. However, to our knowledge, there is no work on evaluating the impact of rate adaptation algorithms in production networks. The reason for this is that all practical production network studies are based on passive monitoring – wireless network clients are autonomous and their configurations are under the control of the users rather than under the control of network administrators, so their active cooperation in network studies cannot be assumed – and there is no way to identify rate adaptation algorithms based on passive traces. Our work on passive trace based 802.11 rate adaptation algorithm identification fills this gap.

There is a large body of work on identifying application layer protocols and algorithms. However, these approaches cannot be easily adapted for the identification of MAC layer algorithms because of the difference in the types of information available at the MAC layer and the application layer. 802.11 rate adaptation algorithms are complex, so an identification method based on explicit enumeration of all algorithm states is difficult if not impossible, suggesting the need for a learning-based approach to the problem.

Our SVM-based approach to 802.11 rate adaptation algorithm identification is based on careful selection of a large number of features from packet traces. Our approach yields a classification accuracy of 95%–100% for 3 out of the 4 algorithms implemented in the popular open-source MadWifi wireless driver. We examine the effect of both feature selection and network dynamics on classification accuracy. We also demonstrate that rate adaptation algorithm identifiers are portable, *i.e.*, classifiers trained under one set of conditions can be used under a different set of conditions. This is an important property because it eliminates the latency of new classifier construction.

7.4 Future Directions

In this section, we discuss directions for future work based on the studies in this dissertation.

1. *Investigating Wireline TCP Throughput Predictor Portability*: In Chapter 4 Section 4.5, we discussed wireline TCP throughput *predictor portability*, *i.e.*, the possibility of using predictors trained on one path for prediction on another path. Portable predictors are desirable because they eliminate the need for training on each new path, *i.e.*, eliminate both the training latency and the measurement traffic introduced, so this is an important direction of future research.

   To facilitate predictor portability, we would have to identify features that are currently implicit in our path-specific predictors. There are two different sets of implicit features. The first set is host-specific features, such as flavors of TCP and various operating system parameters. The second set is path-specific features, such as routes and queuing disciplines. In addition to identifying the implicit features, future work would have to address the issue of determining the feature values using lightweight, practically feasible techniques.

2. *Improving Wireless TCP Throughput Prediction Accuracy*: As discussed in Chapter 5 Section 5.3, while our technique for wireless TCP throughput prediction is practical and has useful accuracy, its accuracy is considerably lower than what we have been able to achieve for wireline throughput prediction. Hence investigating ways to improve prediction accuracy while maintaining our method's other desirable features, such as low latency and practicality, is a useful direction for future work.

   We believe that future efforts to improve prediction accuracy should follow two directions, one for the vehicular client scenario and the other for the stationary or walking node scenario.

   For vehicular clients, a promising direction for future work is augmenting the predictor with information such as the location, trajectory and speed of a vehicular client while it is within range of an AP. The inclusion of such additional information is likely to improve prediction accuracy because [83] has shown that RF behavior patterns for vehicular wireless clients at specific locations persist over time. Including location, trajectory and speed information would also allow prediction of the total volume of data that can be transferred while the vehicular client is within range of an AP. Such a prediction is more useful in the vehicular context than an instantaneous throughput prediction because (a) unlike for stationary clients, instantaneous throughput varies widely for vehicular clients as distance from an AP changes, so the utility of instantaneous throughput is short-lived and (b) knowing how much data can be transferred during a connection will give applications the opportunity to prioritize data transmission and make the most of connection opportunities.

   For stationary or slow-moving clients where the time constraints are not as stringent as for vehicular clients, a promising direction of future work is looking at packet trace based features in greater detail to improve prediction accuracy. Our work on 802.11 rate adaptation algorithm classification, which followed our work on wireless TCP throughput prediction, has shown that a large number of detailed and carefully chosen features yield high accuracy. Based on our experience with rate adaptation algorithm classification, an approach for improving TCP throughput prediction accuracy that suggests itself is the use of fine-grained active probe features such as packet rate and interarrival time in place of coarse-grained probe metrics such as probe throughput.

3. *Toolkit for Automated WLAN Performance Analysis*

   As stated in Chapter 6 Section 6.1, we envision the 802.11 rate adaptation algorithm identification capability being part of a toolkit for automated WLAN performance analysis and debugging.

   Such a toolkit must also provide the capability to determine whether wireless packet loss is due to channel noise or due to interference. Distinguishing between these two causes of packet loss is essential for finding appropriate remedies to wireless network performance problems. Existing methods of distinguishing between the causes of loss are either invasive (*e.g.*, [104] requires modifications to the wireless interface driver, [129] requires a change in the interface between the wireless interface and the driver) or require blanket network monitoring and sophisticated techniques for trace merging and analysis (*e.g.*, [82, 32]). Hence a technique that can identify the cause of packet loss based entirely on passive traces collected using single (or limited) vantage point monitoring would be a valuable contribution.

The requirements of non-invasiveness and single or limited number of monitoring vantage points suggest SVM-based classification of passively collected packet traces as a promising approach for identifying the causes of packet loss[1]. Such an approach would face two main challenges. The first challenge, also faced by the SVM-based approach to 802.11 rate adaptation algorithm identification, is that of identifying appropriate input features and appropriate input feature time scales. 802.11 rate adaptation algorithm identification is *location-agnostic*, *i.e.*, it can be conducted on traces collected at any location as long as the traces contain a reasonable number of packets from the sender of interest. The second challenge is that unlike rate adaptation algorithm classification, loss cause identification may not be location-agnostic. Loss events occur at shorter time scales compared to 802.11 rate adaptation events (rate adaptation events are triggered following at least a handful of packet losses), so they will have to be identified based on smaller numbers of packets. This means that if a monitor misses capturing a few packets, it is likely to hurt accuracy much more for loss cause identification than for rate adaptation algorithm identification. Hence it might be necessary to *(a)* have multiple monitors, and *(b)* place the monitor(s) strategically, *e.g.* close to the sender and/or receiver. If multiple monitors are needed for robustness and high classification accuracy, information from all monitors will have to be combined. However, unlike the inference-intensive explicit trace merging of [82, 32], the SVM-based approach would be able to combine information from multiple monitors implicitly, in the form of additional input features.

[1][38] looks at using SVMs for this problem, however the study is very preliminary and simulation-based.

LIST OF REFERENCES

[1] AirPcap, a wireless packet capture solution for Wireshark. http://www.riverbed.com/us/products/cascade/wireshark_enhancements/airpcap.php.

[2] Description of the Minstrel Algorithm. http://madwifi-project.org/browser/madwifi/branches/madwifi-0.9.4/ath_rate/minstrel/minstrel.txt.

[3] Implementation of the Onoe Algorithm. http://madwifi-project.org/browser/madwifi/branches/madwifi-0.9.4/ath_rate/onoe/onoe.c.

[4] Multi-Class Support Vector Machine. http://svmlight.joachims.org/svm_multiclass.html.

[5] Network Diagnostic Tool (NDT). http://www.measurementlab.net/run-ndt.

[6] Pothole Patrol. http://cartel.csail.mit.edu/doku.php?id=p2_pothole_patrol.

[7] Speedtest. http://speedtest.net.

[8] The Atheros Chipset. http://www.atheros.com/.

[9] The MadWifi Project. http://madwifi-project.org.

[10] Alhussein A. Abouzeid, Sumit Roy, and Murat Azizoglu. Stochastic Modeling of TCP Over Lossy Links. In *The 19th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Tel-Aviv, Israel, March 2000.

[11] Edward G. Allan, Jr., William H. Turkett, Jr., and Errin W. Fulp. Using Network Motifs to Identify Application Protocols. In *Proceedings of the 28th IEEE Conference on Global Telecommunications, GLOBECOM*, Honolulu, Hawaii, USA, November 2009.

[12] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat. A Stochastic Model of TCP/IP with Stationary Random Loss. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, August 2000.

[13] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proceedings of 18th ACM Symposium on Operating Systems Principles, SOSP*, Banff, Canada, October 2001.

[14] Martin F. Arlitt, Balachander Krishnamurthy, and Jeffrey C. Mogul. Predicting Short-Transfer Latency from TCP Arcana: A Trace-based Validation. In *Proceedings of the 5th ACM SIGCOMM Internet Measurement Conference, IMC*, Berkeley, CA, USA, October 2005.

[15] Fan Bai, Daniel D. Stancil, and Hariharan Krishnan. Toward Understanding Characteristics of Dedicated Short Range Communications (DSRC) from a Perspective of Vehicular Network Engineers. In *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Chicago, Illinois, USA, September 2010.

[16] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, and Randy H. Katz. Analyzing Stability in Wide-Area Network Performance. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, Seattle, WA, USA, June 1997.

[17] Paul Barford and Mark Crovella. Critical Path Analysis of TCP Transactions. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Stockholm, Sweden, August 2000.

[18] Yigal Bejerano, Seung-Jae Han, and Li Li. Fairness and Load Balancing in Wireless LANs using Association Control. *IEEE/ACM Transactions on Networking*, 15:560–573, June 2007.

[19] Kristin P. Bennett and Colin Campbell. Support Vector Machines: Hype or Hallelujah? *ACM SIGKDD Explorations Newsletter – Special Issue on Scalable Data Mining Allgorithms*, 2:1–13, December 2000.

[20] Laurent Bernaille and Renata Teixeira. Early Recognition of Encrypted Applications. In *Proceedings of the 8th International Passive and Active Measurement Workshop, PAM*, Louvain-la-neuve, Belgium, April 2007.

[21] Laurent Bernaille, Renata Teixeira, and Kavé Salamatian. Early Application Identification. In *Proceedings of the 2006 ACM Conference on Emerging Network Experiment and Technology, CoNEXT*, Lisboa, Portugal, December 2006.

[22] R. Berwick. An Idiot's Guide to Support Vector Machines. `http://www.svms.org/tutorials/Berwick2003.pdf`, 2003.

[23] Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3), 2000.

[24] John Bicket. Bit-rate Selection in Wireless Networks. Master's thesis, Massachusetts Institute of Technology, February 2005.

[25] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[26] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *The 25th Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Barcelona, Catalunya, Spain, April 2006.

[27] Carsten Burmeister, Ulrich Killat, and B. Bachmann. TCP over Rate-Adaptive WLAN - An Analytical Model and its Simulative Verification. In *Proceedings of the 2006 IEEE Computer Society International Symposium on on World of Wireless, Mobile and Multimedia Networks, WOWMOM*, Buffalo, NY, USA, June 2006.

[28] Carsten Burmeister, Ulrich Killat, and Jens Bachmann. An Analytical Model of Rate-Adaptive Wireless LAN and its Simulative Verification. In *Proceedings of the 3rd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, WMASH*, Cologne, Germany, September 2005.

[29] Vladimir Bychkovsky, Bret Hull, Allen K. Miu, Hari Balakrishnan, and Samuel Madden. A Measurement Study of Vehicular Internet Access Using In Situ Wi-Fi Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Los Angeles, CA, USA, September 2006.

[30] Joseph Camp and Edward W. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. In *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM*, San Francisco, California, USA, September 2008.

[31] Neal Cardwell, Stefan Savage, and Thomas E. Anderson. Modeling TCP latency. In *The 19th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Tel-Aviv, Israel, March 2000.

[32] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, Italy, September 2006.

[33] Pierre Chevillat, Jens Jelitto, A. Noll Barreto, and Hong Linh Truong. A Dynamic Link Adaptation Algorithm for IEEE 802.11a Wireless LANs. In *Proceedings of the IEEE International Conference on Communications, ICC*, Anchorage, AK, USA, May 2003.

[34] Koby Crammer and Yoram Singer. On the Learnability and Design of Output Codes for Multiclass Problems. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, COLT*, Palo Alto, CA, USA, June 2000.

[35] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Traffic Classification through Simple Statistical Fingerprinting. *ACM SIGCOMM Computer Communication Review*, 37(1):5–16, 2007.

[36] Mark E. Crovella and Azer Bestavros. Self-similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE /ACM Transactions on Networking*, 5(6):835–846, 1997.

[37] Robert Daniels and Robert W. Heath. Online Adaptive Modulation and Coding with Support Vector Machines. In *Proceedings of the IEEE European Wireless Conference*, Lucca, Italy, April 2010.

[38] Qianhua Deng and Anni Cai. SVM-based Loss Differentiation Mechanism in Mobile Ad hoc Networks. In *Global Mobile Congress 2009*, October 2009.

[39] Pralhad Deshpande, Xiaoxiao Hou, and Samir R. Das. Performance Comparison of 3G and Metro-Scale WiFi for Vehicular Network Access. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC*, Melbourne, Australia, November 2010.

[40] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery using WiFi. In *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking, MOBICOM*, San Francisco, California, USA, September 2008.

[41] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic Classification Using Clustering Algorithms. In *Proceedings of the ACM SIGCOMM Workshop on Mining Network Data, MineNet*, Pisa, Italy, September 2006.

[42] Vin Sen Feng, Tai Chi Wang, Shih Yu Chang, and Hsi-Pin Ma. Location estimation in indoor wireless networks by hierarchical support vector machines with fast learning algorithm. In *International Conference on System Science and Engineering, ICSSE*, Taipei, Taiwan, July 2010.

[43] Cooperative Association for Internet Data Analysis. `http://www.caida.org/tools`, 2006.

[44] Richard Gass, James Scott, and Christophe Diot. Measurements of In-Motion 802.11 Networking. In *Proceedings of the Seventh IEEE Workshop on Mobile Computing Systems and Applications, WSMCA*, Semiahmoo Resort, Washington, USA, April 2006.

[45] Michael Gastpar and Martin Vetterli. On the Cpacity of Wireless Networks: The Relay Case. In *The 21th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, New York, NY, USA, June 2002.

[46] Alexandre Gerber, Jeffrey Pang, Oliver Spatscheck, and Shobha Venkataraman. Speed Testing without Speed Tests: Estimating Achievable Download Speed from Passive Measurements. In *Proceedings of the 10th ACM SIGCOMM Internet Measurement Conference, IMC*, Melbourne, Australia, November 2010.

[47] Praveen Gopalakrishnan, Predrag Spasojevic, Larry Greenstein, and Ivan Seskar. A Method for Predicting the Throughput Characteristics of Rate-adaptive Wireless LANs. In *Proceedings of the 60th IEEE Vehicular Technology Conference*, September 2004.

[48] Mukul Goyal, Roch Guérin, and Raju Rajan. Predicting TCP Throughput From Non-invasive Network Sampling. In *The 21st IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, New York, NY, USA, June 2002.

[49] Matthias Grossglauser and David N. C. Tse. Mobility Increases the Capacity of Ad-hoc Wireless Networks. In *The 20th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Anchorage, Alaska, USA, April 2001.

[50] Aditya Gudipati and Sachin Katti. Strider: Automatic Rate Adaptation and Collision Handling. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Toronto, ON, Canada, August 2011.

[51] Piyush Gupta and P. R. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[52] David Hadaller, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. Vehicular Opportunistic Communication Under the Microscope. In *Proceedings of the 5th International Conference on Mobile Systems, Applications, and Services, MobiSys*, San Juan, Puerto Rico, June 2007.

[53] Ivaylo Haratcherev, Koen Langendoen, Reginald L. Lagendijk, and Henk J. Sips. Hybrid Rate Control for IEEE 802.11. In *Proceedings of the Second International Workshop on Mobility Management & Wireless Access Protocols, MobiWac)*, Philadelphia, PA, USA, October 2004.

[54] Qi He, Constantine Dovrolis, and Mostafa Ammar. On the Predictability of Large Transfer TCP Throughput. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, PA, USA, August 2005.

[55] Marti A. Hearst. Support Vector Machines. *IEEE Intelligent Systems*, 13:18–28, July 1998.

[56] Gavin Holland, Nitin H. Vaidya, and Paramvir Bahl. A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking, MOBICOM*, Rome, Italy, July 2001.

[57] Wenjie Hu. Robust Support Vector Machines for Anomaly Detection. In *Proceedings of the 2003 International Conference on Machine Learning and Applications, ICMLA*, Los Angeles, CA, USA, June 2003.

[58] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. CarTel: A Distributed Mobile Sensor Computing System. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys*, Boulder, Colorado, USA, November 2006.

[59] Van Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Palo Alto, CA, USA, August 1988.

[60] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, MOBICOM*, San Diego, CA, USA, September 2003.

[61] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of Interference on Multi-Hop Wireless Network Performance. *Wireless Networks*, 11(4):471–487, 2005.

[62] Thorsten Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proceedings of the European Conference on Machine Learning, ECML*, Chemnitz, Germany, April 1998.

[63] Thorsten Joachims. Making large-scale SVM Learning Practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[64] William H. Turkett Jr., Andrew V. Karode, and Errin W. Fulp. In-the-Dark Network Traffic Classification Using Support Vector Machines. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI*, Chicago, IL, USA, July 2008.

[65] Glenn Judd, Xiaohui Wang, and Peter Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys*, Breckenridge, CO, USA, June 2008.

[66] Ad Kamerman and Leo Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, Summer 1997.

[67] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proceedings of the ACM SIGCOMM 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, PA, USA, August 2005.

[68] Anand Kashyap, Samir R. Das, and Samrat Ganguly. A Measurement-Based Approach to Modeling Link Capacity in 802.11-based Wireless Networks. In *Proceedings of the 13th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Montréal, Québec, Canada, September 2007.

[69] Malik Ahmad Yar Khan and Darryl Veitch. Speedo: Realistic Achievable Bandwidth in 802.11 through Passive Monitoring. In *Proceedings of the 3rd IEEE Local Computer Networks Workshop on Network Measurements*, Montreal, Canada, October 2008.

[70] Jongseok Kim, Seongkwan Kim, Sunghyun Choi, and Daji Qiao. CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs. In *25th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Barcelona, Spain, April 2006.

[71] Woojin Kim, Jaemann Park, and H.J. Kim. Target Localization Using Ensemble Support Vector Regression in Wireless Sensor Networks. In *Wireless Communications and Networking Conference, WCNC*, Sydney, Australia, April 2010.

[72] Anurag Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, August 1998.

[73] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. Algorithmic Aspects of Capacity in Wireless Networks. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, Banff, Alberta, Canada, June 2005.

[74] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *MSWiM '04: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2004.

[75] T. V. Lakshman and Upamanyu Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[76] Karthik Lakshminarayanan, Venkata N. Padmanabhan, and Jitendra Padhye. Bandwidth Estimation in Broadband Access Networks. In *Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.

[77] Heeyoung Lee, Seongkwan Kim, Okhwan Lee, Sunghyun Choi, and Sung-Ju Lee. Available Bandwidth-Based Association in IEEE 802.11 Wireless LANs. In *Proceedings of the 11th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, MSWiM*, Vancouver, British Columbia, Canada, October 2008.

[78] Christina Leslie, Eleazar Eskin, and William Stafford Noble. The Spectrum Kernel: A String Kernel for SVM Protein Classification. *Pacific Symposium On Biocomputing*, 575:564–575, 2002.

[79] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of Ad Hoc Wireless Networks. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Rome, Italy, July 2001.

[80] Yi Li, Lili Qiu, Yin Zhang, Ratul Mahajan, and Eric Rozner. Predictable Performance Optimization for Wireless Networks. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Seattle, WA, USA, August 2008.

[81] Dong Lu, Yi Qiao, Peter A. Dinda, and Fabián E. Bustamante. Characterizing and Predicting TCP Throughput on the Wide Area Network. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS*, Columbus, OH, USA, June 2005.

[82] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the MAC-level Behavior of Wireless Networks in the Wild. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, Italy, September 2006.

[83] Ratul Mahajan, John Zahorjan, and Brian Zill. Understanding Wifi-based Connectivity from Moving Vehicles. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC*, San Diego, California, USA, October 2007.

[84] Matthew Mathis, Jeffery Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Computer Communications Review*, 27(3):67–82, July 1997.

[85] Anthony McGregor, Mark Hall, Perry Lorier, and James Brunskill. Flow Clustering Using Machine Learning Techniques. In *5th International Workshop on Passive and Active Network Measurement, PAM*, Antibes Juan-les-Pins, France, April 2004.

[86] Mariyam Mirza, Joel Sommers, Paul Barford, and Xiaojin Zhu. A Machine Learning Approach to TCP Throughput Prediction. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, San Diego, CA, USA, June 2007.

[87] Archan Misra and Teunis J. Ott. The Window Distribution of Idealized TCP Congestion Avoidance with Variable Packet Loss. In *The 18th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, New York, NY, USA, March 1999.

[88] Andrew W. Moore. Support vector machines (tutorial). `http://www.svms.org/tutorials/Moore2001.pdf`, 2001.

[89] Andrew W. Moore and Denis Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, Banff, Alberta, Canada, June 2005.

[90] Andrew Ng. Support Vector Machines (Stanford University CS229 Lecture notes). `http://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf`.

[91] Anthony J. Nicholson, Yatin Chawathe, Mike Y. Chen, Brian D. Noble, and David Wetherall. Improved Access Point Selection. In *Proceedings of the 4th ACM International Conference on Mobile Systems, Applications, and Services, MobiSys*, Uppsala, Sweden, June 2006.

[92] Dragos Niculescu. Interference Map for 802.11 Networks. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC*, San Diego, California, USA, October 2007.

[93] Edgar Osuna, Robert Freund, and Federico Girosi. Training Support Vector Machines: an Application to Face Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Puerto Rico, June 1997.

[94] Jörg Ott and Dirk Kutscher. Drive-thru Internet: IEEE 802.11b for Automobile Users. In *The 23th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Hong Kong, China, March 2004.

[95] Jörg Ott and Dirk Kutscher. A Disconnection-Tolerant Transport for Drive-thru Internet Environments. In *The 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Miami, FL, USA, March 2005.

[96] Teunis J. Ott, J. H. B. Kemperman, and Matt Mathis. The Stationary Behavior of Ideal TCP Congestion Avoidance. `http://www.teunisott.com/Papers/TCP_Paradigm/TCPwindow.pdf`, August 1996.

[97] Jitendra Padhye, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. Estimation of Link Interference in Static Multi-hop Wireless Networks. In *Proceedings of the 5th ACM SIGCOMM Internet Measurement Conference, IMC*, Berkeley, CA, USA, October 2005.

[98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Technologies, Architectures, and Protocols for Computer Communication*, Vancouver, British Columbia, Canada, September 1998.

[99] Javier Del Prado Pavon and Sunghyun Choi. Adaptation Strategy for IEEE 802.11 WLAN via Received Signal Strength Measurement. In *Proceedings of the IEEE International Conference on Communications, ICC*, Anchorage, AK, USA, May 2003.

[100] Vern Paxson. End-to-End Internet Packet Dynamics. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Cannes, France, September 1997.

[101] Ioannis Pefkianakis, Starsky H.Y. Wong, Hao Yang, Suk-Bok Lee, and Songwu Lu. Towards History-Aware Robust 802.11 Rate Adaptation. *IEEE Transactions on Mobile Computing*, to appear, 2012.

[102] John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large Margin DAGs for Multiclass Classification. In *Proceedings of Neural Information Processing Systems, NIPS*, Denver, CO, USA, November 1999.

[103] Lili Qiu, Yin Zhang, Mi Kyung Han, and Ratul Mahajan. A General Model of Wireless Interference. In *Proceedings of the 13th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Montréal, Québec, Canada, September 2007.

[104] Shravan Rayanchu, Arunesh Mishra, Dheeraj Agrawal, Sharad Saha, and Suman Banerjee. Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal. In *The 27th IEEE Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Phoenix, AZ, USA, April 2008.

[105] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Measurement-based Models of Delivery and Interference in Static Wireless Networks. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pisa, Italy, September 2006.

[106] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick G. Duffield. Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, IMC*, Taormina, Sicily, Italy, October 2004.

[107] Bahareh Sadeghi, Vikram Kanodia, Ashutosh Sabharwal, and Edward W. Knightly. Opportunistic Media Access for Multirate Ad Hoc Networks. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Atlanta, Georgia, USA, September 2002.

[108] Charalampos (Babis) Samios and Mary K. Vernon. Modeling the Throughput of TCP Vegas. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, San Diego, CA, USA, June 2003.

[109] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[110] Bernhard Schölkopf, Kashima Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology*. MIT Press, Cambridge, MA, USA, 2004.

[111] Gabriel Gómez Sena and Pablo Belzarena. Early Traffic Classification Using Support Vector Machines. In *Proceedings of the 5th International Latin American Networking Conference, LANC*, Pelotas, Brazil, September 2009.

[112] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and Kimberly C. Claffy. Comparison of Public End-to-end Bandwidth Estimation Tools on High-Speed Links. In *Proceedings of the 6th International Passive and Active Measurement Workshop, PAM*, March 2005.

[113] Alexander J. Smola and Bernhard Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14(3):199–222, 2004.

[114] Joel Sommers and Paul Barford. Self-Configuring Network Traffic Generation. In *Proceedings of the 4th ACM SIGCOMM Internet Measurement Conference, IMC*, Taormina, Italy, October 2004.

[115] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. Improving Accuracy in End-to-end Packet Loss Measurements. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, PA, USA, August 2005.

[116] Joel Sommers, Paul Barford, and Walter Willinger. A Proposed Framework for Calibration of Available Bandwidth Estimation Tools. In *Proceedings of the 11th IEEE Symposium on Computers and Communication, ISCC*, Cagliari, Sardinia, Italy, June 2006.

[117] Augustin Soule, Kavé Salamatian, Nina Taft, Richard Emilion, and Konstantina Papagiannaki. Flow Classification by Histograms: or How to go on Safari in the Internet. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS*, New York, NY, USA, June 2004.

[118] Karthikeyan Sundaresan and Konstantina Papagiannaki. The need for cross-layer information in access point selection algorithms. In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC*, Rio de Janeriro, Brazil, October 2006.

[119] Andrew H. Sung and Srinivas Mukkamala. Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In *Symposium on Applications and the Internet, SAINT*, Orlando, FL, USA, January 2003.

[120] Martin Swany and Rich Wolski. Multivariate Resource Performance Forecasting in the Network Weather Service. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Baltimore, MD, USA, November 2002.

[121] Geza Szabo, Daniel Orincsay, Balazs Peter Gero, Sandor Gyori, and Tamas Borsos. Traffic Analysis of Mobile Broadband Networks. In *Proceedings of the 3rd International Conference on Wireless Internet, WICON*, Austin, Texas, October 2007.

[122] Yuzo Taenaka, Shigeru Kashihara, Kazuya Tsukamoto, Suguru Yamaguchi, and Yuji Oie. Terminal-Centric AP Selection Algorithm based on Frame Retransmissions. In *Proceedings of the 2nd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, PM2HW2N*, Chania, Crete Island, Greece, October 2007.

[123] Arvind Thiagarajan, Lenin S. Ravindranath, Hari Balakrishnan, Samuel Madden, and Lewis Girod. Accurate, Low-Energy Trajectory Mapping for Mobile Devices. In *8th USENIX Symp. on Networked Systems Design and Implementation, NSDI*, Boston, MA, USA, March 2011.

[124] Arvind Thiagarajan, Lenin S. Ravindranath, Katrina LaCurts, Sivan Toledo, Jakob Eriksson, Samuel Madden, and Hari Balakrishnan. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems, SenSys*, Berkeley, CA, USA, November 2009.

[125] Duc A. Tran and Thinh P. Nguyen. Localization In Wireless Sensor Networks Based on Support Vector Machines. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):981–994, 2008.

[126] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[127] Sudarshan Vasudevan, Konstantina Papagiannaki, Christophe Diot, James F. Kurose, and Donald F. Towsley. Facilitating Access Point Selection in IEEE 802.11 Wireless Networks. In *Proceedings of the 5th ACM SIGCOMM Internet Measurement Conference, IMC*, Berkeley, California, USA, October 2005.

[128] Sudharshan Vazhkudai, Jennifer M. Schopf, and Ian T. Foster. Predicting the Performance of Wide Area Data Transfers. In *Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium, IPDPS*, Fort Lauderdale, FL, USA, April 2002.

[129] Mythili Vutukuru, Hari Balakrishnan, and Kyle Jamieson. Cross-layer Wireless Bit Rate Adaptation. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Barcelona, Spain, August 2009.

[130] Shao-Cheng Wang and Ahmed Helmy. BEWARE: Background Traffic-Aware Rate Adaptation for IEEE 802.11. *IEEE /ACM Transactions on Networking*, 19(4):1164–1177, 2011.

[131] Jason Weston and Chris Watkins. Support Vector Machines for Multi-class Pattern Recognition. In *The 7th European Symposium on Artificial Neural Networks, ESANN*, Bruges, Belgium, April 1999.

[132] Starsky H. Y. Wong, Songwu Lu, Hao Yang, and Vaduvur Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Los Angeles, CA, USA, September 2006.

[133] Charles V. Wright, Fabian Monrose, and Gerald M. Masson. On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research*, 6:2745–2769, 2006.

[134] Defeng Xu and Rajive Bagrodia. Impact of Complex Wireless Environments on Rate Adaptation Algorithms. In *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC*, Cancun, Mexico, March 2011.

[135] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Philadelphia, PA, USA, August 2005.

[136] Candy Yiu and Suresh Singh. A Model for Comparing Rate Adaptation Algorithms. In *Proceedings of the Fourth ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, WIN-TECH*, Beijing, China, September 2009.

[137] Sungho Yun and Constantine Caramanis. Multiclass Support Vector Machines for Adaptation in MIMO-OFDM Wireless Systems. In *Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, September 2009.

[138] Sungho Yun and Constantine Caramanis. Reinforcement Learning for Link Adaptation in MIMO-OFDM Wireless Systems. In *Proceedings of the Global Communications Conference, GLOBECOM*, Miami, FL, USA, December 2010.

[139] Jiansong Zhang, Kun Tan, Jun Zhao, Haitao Wu, and Yongguang Zhang. A Practical SNR-Guided Rate Adaptation. In *Proceedings of the 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM*, Phoenix, AZ, USA, April 2008.

[140] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Donald F. Towsley, and Honggang Zhang. Study of a Bus-based Disruption-Tolerant Network: Mobility Modeling and Impact on Routing. In *Proceedings of the 13th Annual International Conference on Mobile Computing and Networking, MOBICOM*, Montréal, Québec, Canada, September 2007.

[141] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pittsburgh, PA, USA, August 2002.

[142] Yin Zhang and Nick Duffield. On the Constancy of Internet Path Properties. In *Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, USA, November 2001.