Enhancement of TCP over wired/wireless networks with packet loss classifiers inferred by supervised learning

Ibtissam El Khayat Pierre Geurts Guy Leduc
Department of Electrical Engineering and Computer Science
University of Liège
Institut Montefiore - B28 - Sart Tilman
Liège 4000 - Belgium
{elkhayat, geurts, leduc}@montefiore.ulg.ac.be

Abstract

TCP is suboptimal in heterogeneous wired/wireless networks because it reacts in the same way to losses due to congestion and losses due to link errors. In this paper, we propose to improve TCP performance in wired/wireless networks by endowing it with a classifier that can distinguish packet loss causes. In contrast to other proposals we do not change TCP's congestion control nor TCP's error recovery. A packet loss whose cause is classified as link error will simply be ignored by TCP's congestion control and recovered as usual, while a packet loss classified as congestion loss will trigger both mechanisms as usual. To build our classification algorithm, a database of pre-classified losses is gathered by simulating a large set of random network conditions, and classification models are automatically built from this database by using supervised learning methods. Several learning algorithms are compared for this task. Our simulations of different scenarios show that adding such a classifier to TCP can improve the throughput of TCP substantially in wired/wireless networks without compromizing TCP-friendliness in both wired and wireless environments.
Keywords: TCP, wireless links, loss cause classification, machine learning

1 Introduction

TCP is the most widely used protocol in the Internet. Its success lies in its reliable transfer and its capacity in avoiding congestion. However, its congestion control, which was developed in the eighties, is based on the fact that packet losses are mainly due to buffer overflows. Hence, it is not adapted to nowadays networks where wireless links are common. Indeed, TCP has no mechanism to distinguish losses caused by link errors from losses caused by congestion, and it reduces systematically its rate whenever it faces a packet loss. This reduction is not justified when there is no congestion and the consequence is that the throughput of TCP over wireless links is lower than what it could be. Several studies have highlighted the bad behaviour of TCP over wireless links (e.g. [33, 39, 41, 3]).

A straightforward solution to increase the throughput of TCP over wireless links is to prevent it from reducing its rate when it faces a loss due to a link error as it does when it faces a congestion event. Two possibilities have been proposed in the literature. The first one consists of hiding link error losses from the sender (for example by splitting the TCP connection [2], or retransmitting in the data link layer). However, such solutions assume some network support. The second approach, which is the one adopted in this paper, consists of endowing one of the end systems with an algorithm that classifies the packet loss causes.

Characterizing analytically the network conditions leading to a certain type of packet loss is difficult because real networks are very complex systems but also because their behaviours depend on a large number of random external factors (e.g., user behaviours, current topologies) which are difficult to model analytically. On the other hand, it is quite easy to simulate the network behaviour (e.g. with a network simulator like `ns-2`[28]) or to gather data from observation of the behaviour of a real network. This is the typical situation where automatic learning techniques are useful. These algorithms are general techniques to extract a model of a system only from data obtained either by direct observations or by simulations of this system. Of interest for our problem are supervised learning algorithms which focus on the approximation of an input/output relationship only from observations of examples of this relationship.

So, we propose here to apply supervised learning algorithms to automatically derive models for discriminating the two possible packet loss causes and then use these models to improve the performance of TCP in wired/wireless networks. The paper is structured as follows. In Section 2, we discuss related works on loss cause classification. In Section 3, we give a short general introduction to supervised learning algorithms. The application of learning algorithms requires the generation of a database from which to infer a model. Section 4 describes how we generate this database. In the same section, we describe the four learning algorithms that are applied to this problem and we evaluate their performance on an independent part of the database. These methods are also compared to the packet loss classifiers proposed in Veno and Westwood. In Section 5, we describe how we propose to enhance TCP with these classifiers and we discuss the different criteria that should be taken into account when choosing a particular classification model. Section 6 evaluates our extension of TCP with these classification models with several simulations. The new protocols are evaluated along essentially three criteria: the gain in wireless links, TCP-friendliness, and bandwidth usage. Finally, we conclude and we give some future work directions.

In order to allow other researchers to compare their classification rules to ours, the database, the TCL code to generate it, the full description of all input variables, and the `ns-2` code of some classifiers are available electronically at [24].

2 Related works

Several papers (e.g., [12, 32, 4, 26, 7, 34]) have studied the problem of the classification of loss causes in wireless networks. Two approaches have been mainly considered. The first one is to rely on the support of the network to perform the classification. For example, TCP-Jersey [40] requires the use of ECN [15] and classifies a loss as due to congestion when there is a congestion notification. The solution in [5] requires the knowledge of the loss rate on the wireless link which could be provided by the base station.

In this paper, we have adopted the second approach that relies only on information available at the end-systems. There are mainly three indicators that can be used at the end-systems to predict loss causes: the inter-arrival times, the one-way delays, and the round-trip-times. These three indicators have been used in the literature. The round-trip-time is used for example in the HMM approach of [26], in Westwood [38], and in NewReno-FF [4]. The inter-arrival times are used by Biaz and Vaidya [7], and the one-way delays by Zigzag [12], Veno [19], and Spike [34]. Among these three indicators, it has been pointed out in [31] and [6] that the round-trip-time alone was not a good indicator of loss causes. Indeed, a modification in the return path affects the round-trip-time without affecting the loss cause. Furthermore, the study in [12] has shown that the inter-arrival times and the one-way delays are complementary when it comes to predict loss causes. So, this suggests that contrary to previous works, we should try and combine all these indicators to design a good classification rule.

In all these works except for [26], the rule is derived in an ad-hoc way and tuned manually using a restricted number of topologies. Westwood [38] classifies a loss as due to a link error when the current round-trip-time is lower than $1.4RTT_{min}$ where RTT_{min} is the minimal round-trip-time estimated since the beginning of the session. The value of 1.4 is the one that yields the best result in their simulations. Veno [19] estimates the backlog by using only the interarrival times and considers that the loss is due to congestion if the backlog is higher than 3. Similar ad-hoc rules are proposed in [7], Spike [34], and Zigzag [12].

The limitations of such approaches are highlighted in [12]. Cen et al. have evaluated the latter three rules in three types of topology with one bottleneck and they have shown that each rule is better than the other two in one of the topologies. They have thus proposed to combine these three rules by first trying to predict the topology of the network and then choosing the most appropriate rule accordingly. However, it is always possible to find a configuration that does not correspond to any of the three topologies considered. Hence, Cen et al.'s rule suffers from the same drawback as the previous ones, even though it is more general.

The approach in [26] is more in the line with the approach adopted in this paper. The classification rule is designed automatically by using a learning algorithm based on Hidden Markov Models. Nevertheless, only one indicator, the round-trip-time, is considered and learning is carried out only on one kind of topology (with one bottleneck, symmetrical one-way and return paths, and fixed flows). Hence, one should expect the learned model to be applicable only in such topology.

In the light of this analysis, our approach extends existing works in two directions. First, instead of using only one indicator, we will combine several of them in our classification rules. Second, and more importantly, the rules will be tuned automatically by supervised learning techniques from a large database of losses observed by simulation in very diverse and randomized network conditions. In this way, we will hopefully obtain a rule that will work as well as possible *in average*

instead of the best possible rule for a specific network topology though inadequate in most cases. Furthermore, to verify the good average behaviour of our rules, we will evaluate them at classifying loss causes in another large number of randomized topologies not seen during the learning stage. Such large-scale evaluation will certainly highlight the feasibility of the approach.

Let us also outline the passive nature of our classifier. An alternative approach would be to adopt a probing scheme like in TCP-Probing [35]. In this protocol, when a data segment is lost, the sender tries to identify the cause by initiating a probe cycle (during which data transmission is suspended) to monitor network conditions. By comparing the measured RTTs of probes the sender determines the level of congestion and TCP will react accordingly. This contrasts with our approach, which tries to infer the loss cause without probing.

It is also important to note that our goal is to explore a design path that consists in adding a classifier to TCP without changing its congestion control algorithm, nor its error recovery algorithm. In our proposal, a loss classified as due to a link error will simply be ignored by the congestion control algorithm, and retransmitted in the usual way. Other papers have instead focused on the improvement of those algorithms (e.g. TCP-Westwood [38] and TCP-Veno [19]). Another example is proposed in [37] where the dynamics of TCP in the context of wireless/mobile networks is studied, with the goal of improving smoothness without damaging responsiveness.

Therefore, even though we only address the loss classification problem in this paper, we have to keep in mind that improving TCP in heterogeneous wired/wireless networks and mobile networking has multiple facets. In [36] the authors discuss a general framework to address TCP design principles in this context. Among other things, the paper discusses the requirements of error detection and proposes related error recovery techniques.

3 Supervised learning

Automatic learning denotes methods which aim at extracting a model of a system (in our case, a computer network) from the sole observation (or the simulation) of this system in some situations. By model, we mean some relationships between the variables used to describe the system. The goal of this model may be to predict the behaviour of this system in some unencountered situations or to help understanding its behaviour.

Supervised learning is the part of automatic learning which focuses on modeling input/output relationships. More precisely, the goal of supervised learning is to identify a mapping from some input variables to some output variable on the sole basis of a sample of observations of these variables. Formally, the sample of observations is called the learning sample LS and is a set of input/output pairs, $LS = \{< x_1, y_1 >, < x_2, y_2 > , \ldots, < x_N, y_N >\}$, where x_i is the vector of values of the input variables (also called the attributes) corresponding to the ith observation (also called an object) and y_i is its output value. Attribute values may be discrete or continuous. The goal of supervised learning can be formulated as follows: From a learning sample LS, find a function $f(x)$ of the input attributes that predicts at best the outcome of the output attribute y for any new unseen values of x. When the output takes its values in a discrete set $\{C_1, C_2, \ldots, C_m\}$, we talk about a classification problem and when it is continuous, we talk about a regression problem.

This problem is solved by a (supervised) learning algorithm. Loosely speaking, a learning algorithm receives a learning sample and returns a function f (an hypothesis or a model) which is chosen in a set of candidate functions (the hypothesis space). There exist many learning algorithms, which differ mainly in the hypothesis space but also in the optimization algorithm that searches this space for a good model. Among the most popular supervised learning algorithms, there are decision trees and neural networks.

The main criterion used to assess learning algorithms is their prediction accuracy, i.e. the way the model they produce generalizes to unseen data. Usually, the ranking among algorithms depends largely on the problem and how well the basic hypotheses of the learning algorithm are satisfied by this problem. Another important criterion is the interpretability of the learning algorithm, i.e. if it produces comprehensible models or not. A third criterion is the computational efficiency of the method, i.e. the time needed to learn a model but also the time to apply this model to make new predictions. In general, there is a tradeoff between these three criteria. An accurate method is likely to be resource demanding or induces non comprehensible models, while interpretable models are often not very accurate. Also, the relative importance assigned to each criterion is highly application-dependent. This makes existing algorithms complementary and none of them can be claimed to be globally superior to all other ones.

In this paper, we will make use of three different families of learning algorithms that are briefly described in Section 4.2. For a complete reference on supervised learning algorithms, see for example [13] or [23].

4 Loss classification by supervised learning

In this section, we focus on the problem of the derivation and the evaluation of a model for predicting loss causes by using supervised learning techniques. The question of the application of this model to improve TCP performance will be addressed in the next sections.

4.1 The Database

To solve our problem of losses classification, each observation $< x_i, y_i >$ of our learning sample will be an input/output pair where the inputs x_i are some variables that describe the state of the system at the occurrence of a loss and the (discrete) output y_i is either C to denote a loss due to congestion or LE to denote a loss due to a link error.

To make the model generally applicable, the observations in the database must be as much as possible representative of the conditions under which we will apply the classification model. So, the database generation should take into account all the uncertainties we have a priori about the topology of the networks, the user behaviours, and the protocols. The way we generated our observations is described in Section 4.1.1. Another important question is the choice of the input variables which is discussed in Section 4.1.2.

4.1.1 Database generation

The database was generated by simulations with the network simulator `ns-2`[28]. To generate our observations of losses, we have used the following procedure: a network topology is generated randomly and then the network is simulated during a fixed amount of time, again by generating the traffic randomly. At the end of the simulation, all losses that have occurred within this time interval are collected in the database. This procedure is repeated until we have a sufficient number of observations in the database. In practice, the larger the learning sample, the better it is for supervised learning algorithms. In our study, we have collected 35,441 losses that correspond to more than one thousand different random topologies. This large diversity of scenarios is necessary to ensure the generality of the obtained rules.

To generate a random topology, we first select a random number of nodes (between 10 and 600) and then choose randomly the connections between these nodes. The bandwidth, the propagation delay and the buffer size of the links were chosen randomly. The bandwidth is chosen between 56Kb/s and 100Mb/s while the propagation delay varies between 0.1ms and 500ms. As Droptail is the most widely deployed policy [16], our simulations all use this latter policy.

The number of wireless links, their place in the topology, the error model and the loss rate were also drawn at random. The error models are either the simple uniform error model, to mimic random losses, or the two-state Gilbert-Elliott model, to mimic bursty losses. These two models are often used to simulate wireless losses (eg. [38], [21]).

Concerning the traffic, 60% of the flows at least were TCP flows[1] and the others were chosen randomly among TCP and other types of traffic based on udp and proposed by `ns-2`. The senders, the receivers, and the duration of each traffic were set randomly. Thus, the database contains losses belonging to short and long TCP sessions. This random choice of traffic length allows us to avoid making any assumption about the network load which is randomized in the database.

The randomness of all network conditions may be questioned, but it has two intrinsic merits. Firstly, it creates no bias, which could otherwise be criticized. Secondly, the existence of possibly unrealistic topologies and flows in our learning set can only reduce the accuracy of our learning algorithms. So, we can consider our learning set as an unbiased worst case.

4.1.2 The choice of the inputs

The choice of the input variables is directed by several constraints. Of course, it should be possible to predict a congestion event from the observation of these variables. For our classification models to be practically useful, these variables should also be measurable (either at the receiver or at the sender side). The supervised learning method should also be taken into account when choosing the inputs. Some learning algorithms, like for example decision trees, are, at least to some extent, able to distinguish automatically relevant variables from irrelevant ones for a specific problem. However, other learning algorithms, like neural networks, suffer more in the presence of useless variables and, hence, it is always preferable to restrain the set of inputs as much as possible.

At the end system, the information we can measure to predict congestion is the inter-arrival times and (the variations of) the one-way delay (RFC1323). Subsequently we will also use the term queuing delay to denote the one way delay. These measures can be obtained at both sides. Indeed, to compute these values,

[1] We have chosen Newreno but any other version of TCP could be used.

a host only needs the instant the packet was sent and the instant it was received. At the receiver side, the instant when the packet is received is known and the moment when the packet was sent is the timestamp of the packet it received. At the sender side, the instant when the packet was sent is carried in the acknowledgment and the instant the packet was received is the timestamp of the acknowledgment. The one-way delay is then the difference between the timestamp of the acknowledgment and the timestamp of the TCP packet, and is actually the real one-way delay minus the difference between the clocks of the sender and the receiver. This difference is not important in our study since we will see below that our inputs are based only on relative variation of the one-way delay.

To compute our inputs, we use the information provided by the three packets following the loss and the packet that precedes it.[2] We could choose to include in our inputs directly the values of the inter-arrival times and the one-way delay for these packets. However, this will make the model dependent on absolute values of these measures, which is not a good idea if we want our models to be as much as possible independent of the particular network conditions. So, instead, we propose to use as inputs only relative values of these measures normalized in different ways.

To this end, we further compute the average, the standard deviation, the minimum, and the maximum of the one-way delay and inter-arrival times for the packets that are sent during one round-trip-time and we maintain these values at each time for the last two round-trip-times before the current time. Our final inputs are then various functions (about 40) relating these (8) values at the occurrence of a loss to the inter-arrival times and the one-way delays of the three plus one packets surrounding the loss.

To give some examples of the inputs, let us define the following four groups of packets: the packets for the two preceding round-trip-times, the packet before the loss and the three packets after the loss. Then, for both indicators (i.e. inter-arrival times and one-way delay), we include in the inputs the ratios between the averages, the minima, and the maxima of this indicator in all pairs of groups. We also consider the ratio between the maximum in one group and the minimum in another group of each indicator. Denoting by x the value of one of the two indicators for the packet either after or before the loss, the following inputs were also included: $\frac{x-\mu}{\sigma}$, where μ and σ are the average and standard deviation of the same indicator in one of the two groups corresponding to the round-trip-time.

We have also introduced among the inputs the number of losses, which in most TCP releases can be obtained only at the receiver side. Nevertheless, our experiments show that this input is not important for the classification of packet loss. So, since all other inputs can be computed indifferently at both sides, the classification of losses can be carried out either by the sender or by the receiver.

The complete description of all input variables is available in [24].

4.2 Supervised learning methods

We have chosen to compare here three different families of learning algorithms with quite different characteristics. We give below only a brief description of these algorithms. We refer the interested readers to the corresponding references for further details.

Decision trees [10]. This is one of the most popular learning algorithms. A decision tree represents a classification model with a tree where each interior node is labeled with a binary test based on one input attribute and each terminal node is labeled with a value of the output (here C or LE). Figure 2 shows the top of one decision tree that was built for this problem. To classify an observation with such a tree, we simply propagate it from the top node to a terminal node according to the test issues and the prediction for this observation is the value associated with the terminal node. A decision tree is built automatically from the learning sample in a greedy top-down fashion. The algorithm starts from the whole learning sample that it recursively splits with binary tests. A score measure is defined to evaluate how well a test is able to separate observations of different classes in a node and the test that maximizes this score measure is selected at each node. The splitting of a node is stopped when the output is constant in this node or some stopping criterion is met (e.g. the size of the local subsample goes below some threshold or the split is deemed non significant according to some statistical test). The two main advantages of decision trees with respect to other learning algorithms is their readibility (by construction) and their computational efficiency (both for learning the tree and for testing it). In our experiments, we have used the algorithm for decision tree induction proposed in [10].

Tree bagging [9] and tree boosting [18]. Although they present several nice characteristics, decision trees are often not competitive with other learning methods in terms of accuracy. In supervised learn-

[2] Since the variation in the down-link has no effect on the losses undergone on the up-link we do not consider the one-way delay and the inter-arrival times of the acknowledgment.

ing, ensemble methods are generic techniques that improve a learning algorithm by learning several models (from the same learning sample) and then by aggregating their predictions. In our experiments, we will use two ensemble methods with decision trees: bagging and boosting. Bagging builds each tree of the ensemble from a bootstrap sample[3] drawn from the original learning sample. With boosting, the trees are built in sequence. Each tree is built by increasing the weights of the learning sample cases that are misclassified by the previous trees of the sequence (the exact weight update formulas can be found in [18]). In both cases, the prediction of the ensemble of trees for a new observation is the majority class among the classes predicted by all trees for this observation. Both methods often improve very importantly the accuracy of decision trees. However, since they require to build T decision trees instead of one (in our experiments, T was fixed to 25), the computing times and the memory required to store a model is T times higher than for a classical decision tree.

Multilayer perceptrons [8]. Multilayer perceptrons are a particular family of artificial neural networks. Neural networks represent a model as the interconnection of several small units called perceptrons that compute a weighted average of their inputs and send this average through a non linear functions (usually a hyperbolic tangent). This method usually gives more accurate models than decision trees but a neural network is not interpretable and is also much more demanding in terms of computing times and computer resources. In our experiments, we have used a Levenberg-Marquard optimization algorithm to learn neural networks and we have tried several neural network architectures. Only the best results are presented.

Of course, we have also tested other learning algorithms on our problem, but a preliminary study [20] has shown that the selected ones provide the best results in this case. For our experiments in this paper, we have used a data mining software called PEPITo[4], as well as our own implementation of bagging and boosting.

4.3 Model evaluation

Usually, the error rate of the model at classifying loss causes in the learning sample is very small since the learning algorithm explicitly tries to minimize this error. Thus, this error is not a good indication of the ability of the model at classifying losses in new unseen topologies. To get a more reliable estimate of the error of the classifier, we have thus randomly divided the database into two parts: a learning sample that is used to learn the model and a test sample on which the resulting classifier is tested. Since the losses in both samples are obtained from different topologies, the error rate of the model at classifying losses in the test sample gives a good idea of the probability of misclassification of our model in a new situation.

When evaluating a model, there are two errors of interest: the probability that the model misclassifies a congestion event as a link error and the probability that it misclassifies a link error as (an error due to) a congestion event. We will denote these errors respectively Err_C and Err_{LE}. Of course, it is important to minimize these two types of errors but we will show later that if we want to provide TCP-friendliness for example, we need especially to minimize errors on congestion. So, independently of the application of the model, it is very desirable to be able to favour the accuracy of the prediction of one type of loss over the other.

Actually, all learning algorithms chosen here not only provide a class prediction for each value of the inputs x but also provide an estimate of the probability of each class, C or LE, given x, i.e. two numbers $\hat{P}(C|x)$ and $\hat{P}(LE|x)$ such that $\hat{P}(C|x)+\hat{P}(LE|x) = 1$. The class given by the model is then LE if $\hat{P}(LE|x)$ is greater than a threshold P_{th} and C otherwise. By default, the value of P_{th} is fixed to 0.5 so as to treat each class equally. However, by changing the threshold, we can easily favour the accuracy of the prediction of one class over the other. By taking P_{th} lower than 0.5, more losses will be predicted as due to link error and hence, we will decrease Err_{LE} and increase Err_C. On the opposite, by taking P_{th} greater than 0.5, we will decrease Err_C and increase Err_{LE}. So, this parameter allows us to obtain different classification models with different tradeoff between the two types of errors. It is also important to note that, whatever the classification model, the user can choose the tradeoff that fits ones application without re-running the learning algorithm. All one has to do is to change the value of P_{th} when making a prediction with the model.

In the next section, we will evaluate classification models with different values of P_{th}.

4.4 Results

The database of 30,441 losses has been randomly divided into a learning sample of 25,000 cases and a test sample with the remaining 10,441 cases. A classification model is built with each method on the learning

[3] A bootstrap sample from a learning sample LS of size N is obtained by sampling N observations uniformly and with replacement from LS.

[4] http://www.pepite.be

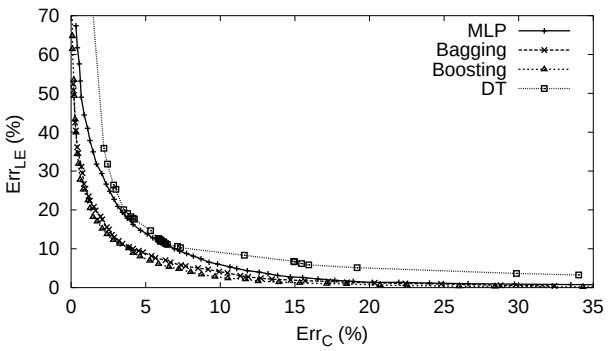

Figure 1: The classification errors obtained by each method when varying P_{th}

Table 1: Comparison of the different learning algorithms.

Method	AUC	Time (μsec)
DT	0.9424	1
MLP	0.9761	144
Bagging	0.9796	37
Boosting	0.9840	33

sample and its two types of errors Err_C and Err_{LE} are evaluated on the test sample for different values of P_{th} going from 0.02 to 0.98 by step of 0.02. Figure 1 plots Err_{LE} in function of Err_C when varying P_{th} for the four methods. The best models on this graph are those which are the closest to the origin.

The results are quite good, especially considering the diversity of the network topologies represented in the test sample. The curves clearly show that there is a tradeoff between the two types of error. It is possible to reduce the error of one class to zero but this is always at the expense of the other. One important thing to note is that we can decrease greatly the error on the classification of LE without increasing too much the error on the detection of congestions. We also observe that the curves are not symmetrical: we have more errors over LE than over C. This could be explained by the fact that an error due to a link error could happen just before a congestion event. In this case, it is impossible to distinguish this error from an error really due to congestion and this loss will be most probably misclassified as a congestion loss. So, this certainly contributes to the fact that there are more errors on the classification of link errors than on congestions.

By flipping the curves of Figure 1 with respect to the axis $Err_{LE} = 50\%$, we get exactly the so-called ROC curves (Receiver Operating Characteristic) often used in the machine learning literature [14]. The area under this curve, denoted AUC, is a popular measure to compare the quality of different methods: the higher the area, the better the method. Table 1 reports this values for the different learning algorithms. For comparison, a method that would guess the loss cause at random would have an AUC of 0.5. All classifiers are much better than random guess. Among them, the decision tree is certainly the worst one. The Neural network is better (especially for the extreme values of P_{th}). Bagging and boosting are very close to each other and also consistently better than the other two methods.

For comparison, we have also evaluated the rules of Veno [19] and Westwood [38] over our test set of 10,441 losses. We obtained respectively $(Err_C, Err_{LE}) = (54.29\%, 0.50\%)$ and $(Err_C, Err_{LE}) = (63.20\%, 4.25\%)$. The point corresponding to TCP, which has no mechanism to distinguish loss causes, is $(Err_C, Err_{LE}) = (0, 100\%)$. These results are much worse than what we obtain with our approach. For example, for the same value of Err_{LE} as Veno and Westwood, boosting gives respectively an error Err_C of about 22% (for $P_{th} = 0.18$) and 8% (for $P_{th} = 0.4$). This shows that our rules are much better than these two latter rules in average over a large number of random topologies even though the Veno's or Westwood's rules can sometimes be better in some specific network conditions.

Another important criterion to compare learning algorithms is the computing times needed to make a classification. To give a rough idea of the requirement of each classifier, the last column of Table 1 gives for each method the average time[5] needed to classify one loss from the test sample, assuming the inputs have been computed. The decision tree is by far the fastest model. The neural network is the slowest model and bagging and boosting are more than 30 times slower than the decision tree. In terms of the computer resource needed to store the model, the decision tree is again the cheapest model while boosting and bagging multiply the requirement by about 25 (the number of trees in the ensemble).

4.5 Interpretability of decision trees

Even if decision trees are not the best method in terms of accuracy, one of its advantages is that it provides interpretable rules. So, it is interesting to try and explain the tests that appear at the top of the tree. Figure 2 shows the first three levels of the tree that was built

[5]The classifier is implemented in C and runs on a Pentium 4 2.0 GHz

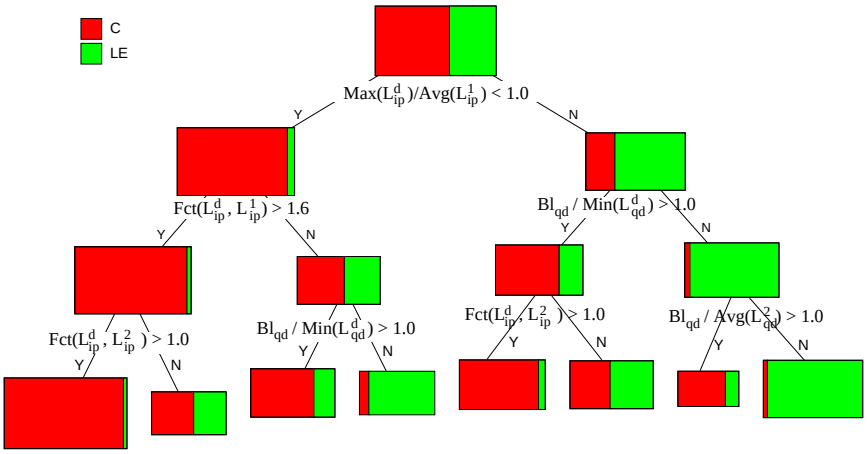

Figure 2: Top of a decision tree built for this problem

Table 2: The notations used

L^d	list of the 3 packets following the loss (d is due to the fact that these packets generate duplicates ack)
L^1	list of the packets received during the previous rtt
L^2	list of the packets received during the rtt preceding the previous rtt
Bl	the packet received just Before the Loss
Bl_{qd}	the queuing delay of Bl
$L^X_{ip}, X \in \{d, 1, 2\}$	inter-arrival times of the L^X packets
$L^X_{qd}, X \in \{d, 1, 2\}$	queuing delays L^X packets
Avg, Max, Min	respectively average, maximal and minimal
$Fct(X, L)$	is equal to $\frac{Avg(L)-Avg(X)}{Std(L)}$

from our database. Tests from deeper nodes are used to refine the classification. The notations used in the tests are described in Table 2. The colour in the rectangular box reflects the proportion of the two types of losses among the losses that reach this node.

The first test is $[Max(L^d_{ip})/Avg(L^1_{ip}) < 1]$ and when it is true, the loss is more probably due to congestion (the left box is mainly dark grey). In fact, if $Max(L^d_{ip}) < Avg(L^1_{ip})$, the packets received after the loss arrive in a more "confined" manner than during the round-trip-time preceding the loss. If the packets are more "confined", then it is very likely that the queue is long and it is not surprising that the loss is most probably due to congestion.

The test $Bl_{qd}/Min(L^d_{qd}) < 1 \Rightarrow$ LE, which appears at two positions, is also very intuitive in the case of one flow and one bottleneck. Let us explain it in this case. In fact, if the loss is due to a congestion event, then the packet preceding the loss is in the end of the buffer and then its queuing delay (Bl_{qd}) is the highest one a packet can have. So if after a loss, one packet can have a queuing delay longer than Bl_{qd}, then the loss was not due to a congestion event but to a link error.

The test $Bl_{qd}/Avg(L^2_{qd}) < 1 \Rightarrow LE$ is comparable to the previous test.

The other tests include the two variables $Fct(L^d_{ip}, L^2_{ip})$ and $Fct(L^d_{ip}, L^1_{ip})$, which are in fact very similar. The values 1 and 1.6 are difficult to explain but the idea behind these tests is that if the average of the inter-arrival times following the loss is "too" far (i.e. low in this case) from the average of this indicator one round-trip-time or two round-trip-times before, then the packets are more "confined". Thus, again, it is very likely that a queue was growing and the loss is most probably due to congestion.

Note that our aim in this section is only to show that the rules, although they are produced automatically by a computer program from the set of random simulations, are to a certain extent interpretable. However, the reader should be aware that these rules only give a very partial picture of the whole decision tree and the way a packet loss is classified is much more complex than the simple tests illustrated here. This is even more so for the boosting model that combines several trees.

5 Choosing the best method to improve TCP performance

The way we propose to enhance TCP (Newreno) with a classification model is the following: Each time a loss is detected by a triple duplicate, its cause is determined by the classification model. If the loss is classified as due to congestion, the sender proceeds as usual (i.e. it divides its congestion window by two). Otherwise, it maintains its congestion window constant. Subsequently, we will call this protocol TCP+classifier.

At this point, we have at our disposal several classification models corresponding to different learning algorithms and different tradeoffs between the two types of error (by changing P_{th}). So, the question is now which particular combination should be used to improve TCP performance in wireless case without losing TCP-friendliness. In this section, we discuss the different constraints that must be taken into account when choosing a particular classification model.

5.1 Practical implementation of the protocol

The computing times to make a prediction and the computer resource needed to store the model are not negligible in most cases. What we can afford according to these criteria will depend on which side we decide to put the classification algorithm. If the sender is in charge of the classification, using a relatively complex classification algorithm could be a problem if the sender has a great number of clients (which is usually the case when the sender is a server). So, it could be desirable to trade some accuracy for more efficiency by using a decision tree instead of boosting. Of course, if the sender has a benefit in providing the highest rate to the receivers, or if it has only a few clients (e.g. proxy), it may still prefer to use boosting. On the other hand, if the receiver is in charge of the classification, the complexity of the algorithm will not be an issue anymore and there is no reason to avoid using the boosting model. However, once a loss is classified, the receiver will need to send the classification result to the sender or at least a message where it asks the sender not to decrease its rate. The problem is that by allowing this kind of exchange, we will open greatly the door to distrusted receivers.

This discussion shows that there is an interest even in less accurate models, for computational efficiency reasons. So, to make our experiments independent of the particular choice of implementation, we will evaluate both the decision tree and the boosting classifier, the first model because it is the fastest one and the second model because it is the most accurate.

5.2 The effect of misclassifying congestions

Once a classification method is chosen, we have also to choose the tradeoff between the two types of error, i.e. a value of P_{th}. One important criterion to select this value is the constraint that our protocol should be TCP-Friendly([17], [27]). The notion of TCP-Friendliness has been defined in the context of wired networks but this definition can be extended to the case of wireless networks. A TCP-Friendly protocol in wireless network is a protocol that allows TCP to have a throughput similar to the one it would get if it were in competition with another TCP in similar conditions.

If a TCP+classifier misclassifies quite often its congestion losses in wired networks, it will be unfair to TCP since it will force TCP to decrease its rate to avoid congestion. And if it is in competition with similar traffics, then a global misclassification will lead the network to a blackout, followed by timeout expirations. Thus, instead of reacting to triple duplicates like TCP would, TCP+classifier would wait for the timeout expiration and thus get in average less throughput than a normal TCP.

To demonstrate this idea, we have augmented TCP Newreno with the classification rule of Veno [19], referred to as TCP+Veno-classifier, which misclassifies many congestion losses ($Err_C = 54.29\%$). We have simulated a scenario where TCP is in competition with such a TCP+Veno-classifier and we get the behaviour presented at the top of Figure 3. Unsurprisingly, the TCP+Veno-classifier, which reduces its window size less often than TCP, gets much more bandwidth than the latter. Moreover, the lower part of the same figure shows on one hand the throughput of an aggregate of TCP+Veno-classifiers competing over one link, and on the other hand another simulation with an aggregate of TCP flows in similar conditions. This time, the standard TCP aggregate gets much more bandwidth, because all flows better control congestion.

So, for their own benefit and the benefit of the community, a host should not use a model that makes many errors on the detection of losses due to congestion. Thus, in other words, high enough values of P_{th} should be preferred when choosing a classification model.

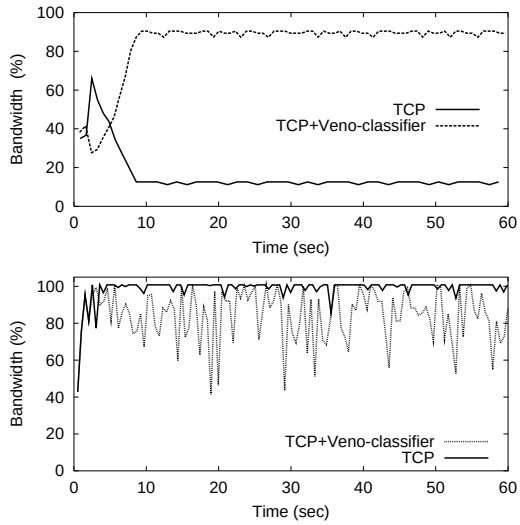

Figure 3: The effect of misclassifying congestion losses. Top: A TCP+Veno-classifier vs TCP. Bottom: Link utilization by an aggregate of TCP+Veno-classifiers, compared to the link utilization an aggregate of standard TCPs would get

5.3 The maximum classification error on congestion still offering TCP-friendliness

The previous section has shown that to preserve TCP-Friendliness and good usage of the bandwidth, the misclassification of congestion should not be too high. It is clear that if the misclassification of congestion remains close to zero, the TCP+classifier will be TCP-friendly. The higher Err_C, the lower Err_{LE}, and thus the higher the gain in wireless case. But the question is how far from zero can we go without losing TCP-Friendliness. In this section, we try and find analytically the maximal value of Err_C for which we still have TCP-friendliness. This bound will allow us to fix the value for P_{th} for a given classifier.

If TCP-Friendliness is provided in the case of wired networks, it will be provided in the case of wireless networks. So, we will try and find the maximum error preserving TCP-Friendliness only in the case of wired networks. To provide TCP-Friendliness, the TCP+classifier should have a throughput belonging to $[1/K B_{tcp}, K B_{tcp}]$ with $K \leq 1.78$ [17] and where B_{tcp} is the throughput of TCP following the same path as the TCP+classifier.

Suppose that we have one TCP and one TCP+classifier following the same path, passing through the network, which is considered as a black box, and that both traffics lose a proportion p of their packets. The TCP+classifier is a normal TCP except that it reacts only to a proportion $p.(1-Err_C)$ of packets instead of p in the case of a normal TCP. So, according to Padhye et al. [30], the throughput of the TCP classifier, if we assume that there is no delayed acknowledgment ($b = 1$)[6], is equal to:

$$B_C = \frac{1}{RTT\sqrt{\frac{2pY}{3}} + T_0 min(1, 3\sqrt{\frac{3pY}{8}})pY(1+32p^2Y^2)}$$

where $Y = 1 - Err_C$, while the one of TCP is equal to

$$B_{tcp} = \frac{1}{RTT\sqrt{\frac{2p}{3}} + T_0 min(1, 3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

The goal is to find the maximal value for Err_C such that

$$F(Err_C, p, RTT, T_0) = \frac{B_C}{B_{tcp}} < K$$

It can be shown that F, over the definition domain(i.e. $RTT, T_0 \geq 0$ a,d $p, Err_C \in [0,1]$), monotonically decreases with RTT and increases with T_0. Hence, if $F_{RTT\to 0} \leq K$ and $F_{T_0\to+\infty} \leq K$, we have necessarily

$$F(Err_C, p, RTT, T_0) < K \ \forall RTT, T_0, p$$

Furthermore, $F_{RTT\to 0} = F_{T_0\to+\infty} = F(Err_C, p, 0, \infty)$, and $F(Err_C, p, 0, \infty)$ can be decomposed into:

$$\begin{cases} \frac{(1+32p^2)}{\sqrt{1-Err_C}^3(1+32p^2(1-Err_C)^2)} & \text{if } 3\sqrt{\frac{3p}{8}} \leq 1 \\ \frac{(1+32p^2)}{(1-Err_C)(1+32p^2(1-Err_C)^2)} & \text{if } 3\sqrt{\frac{3p(1-Err_C)}{8}} \geq 1 \\ \frac{(1+32p^2)}{3\sqrt{\frac{3}{8}p(1-Err_C)}(1+32p^2(1-Err_C)^2)} & \text{otherwise} \end{cases} \quad (1)$$

The curve of F corresponding to the last condition is under the one corresponding to the second condition for all values of p (such as $3\sqrt{3p/8} > 1$) and Err_C. Hence, it is sufficient to bound F for the first two conditions for all p and Err_C to globally bound F. The conditions $3\sqrt{3p/8} \leq 1$ and $3\sqrt{3p/8} \geq 1$ are equivalent respectively to $p \leq 8/27 \simeq 0.3$ and $p \geq 0.3$.

Figure 4 shows these two curves in function of Err_C and p. We can see that to keep a ratio between the throughputs of TCP classifier and TCP below 1.78, we need to keep Err_C below 18%. This value is much lower than the value of Err_C for both Veno and Westwood.

It is important to note that the functions corresponding to the first two conditions of Eq. (1) are both

[6]Using $b = 1$ in the formula was recommended in [22]

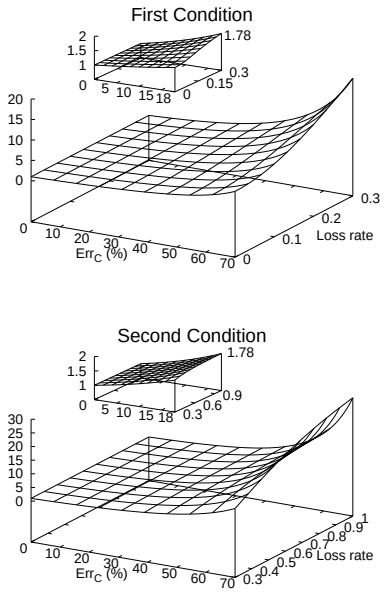

Figure 4: The graph corresponding to $F(Err_C, p, 0, \infty)$

increasing with p. So it is sufficient to find Err_C that makes K greater than F corresponding to the maximum plausible value of p.

6 Simulations with the learned models

We have chosen to restrict our experiments to the models given by the decision tree method and the one given by boosting. The decision tree is chosen for its low complexity which makes it usable at the sender side, and the second method is chosen for its accuracy. Boosting gives the best results among the other methods. It can be implemented at the receiver side or at the sender side if the computational cost is less an issue.

From the previous discussions, it is clear that the characteristics of a TCP+classifier will greatly depend on the chosen value of P_{th}. So, we test here both methods with different values of P_{th} that provide errors on congestion slightly lower but also greater than the analytical bound of 18% determined in the previous section. More precisely, for DT, we use the parameter values P_{th}=0.5, 0.08, and 0.04 corresponding respectively to $(Err_C, Err_{LE}) = (6.0\%, 12.09\%)$, $(15.93\%, 5.86\%)$ and $(29.85\%, 3.65\%)$ on Figure 1. We also compare the results given by boosting with the parameter values equal to 0.5, 0.25, and 0.2. These values correspond to the points $(Err_C, Err_{LE}) = (4.63\%, 8.03\%)$, $(15.32\%, 1.25\%)$, and $(20.71\%, 0.68\%)$ on the same figure.

For comparison, we test also the classification rule used by Veno. We have chosen Veno because its two errors are lower than the ones of Westwood. Note that we have only used the classification rule of Veno, not the whole protocol. Indeed, Veno and Westwood react in a more sophisticated way when a packet loss is classified as due to a link error. In this paper, we have only tested the effect of the accuracy of the classification.

In this section, we evaluate the classification models in wireless and wired networks along three criteria: the gain obtained in wireless links, the fairness towards TCP, and finally the bandwidth usage. The last criterion is less important than the others but it can still motivate a choice or another. All the experiments have been done with `ns-2`. The `ns-2` code for the two classifiers can be found at [24].

6.1 The improvement in lossy links

We test the topology used in [38] and illustrated in Figure 6. It represents a hybrid network. The first part, (Sender-BS), is wired, and the second part, which connects the base station to the sink, is wireless. The bottleneck is the wireless link, which has a bandwidth equal to 11Mb/s. To have a good point of comparison, we run also simulations with an artificial TCP that classifies perfectly the cause of losses detected by triple duplicates.

We compare the throughput with each classification model when we vary the packet loss rate from 0 to 5% over the wireless link. Each simulation is run 50 times. Figure 8 illustrates the throughput obtained by a perfect and a normal TCP, and also the throughput obtained by the classifiers we have tested. The most important thing to note is that some TCP+classifiers can get throughput very close to the one of the perfect TCP. These models are Boosting-0.25 and Boosting-0.2. Veno is not far from them. In the middle, we have DT with parameter 0.04 which doubles the throughput of TCP when the loss rate becomes high (i.e. $\geq 3\%$).

We can see however that all methods improve TCP even if some of them improve it only slightly. The gains we obtain with DT-0.5 and Boosting-0.5 are especially low in this scenario, but they give higher gain in other scenarios. In fact the gain depends on a lot of factors. Among them we have the congestion window sizes, which reduces the gain when it is small. Actually, with very small congestion window size, TCP will be more often in the slow start phase and the losses will be detected by timeouts and not by triple duplicates.

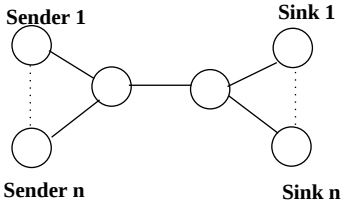

Figure 5: Topology 1

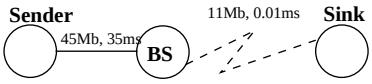

Figure 6: Topology 2

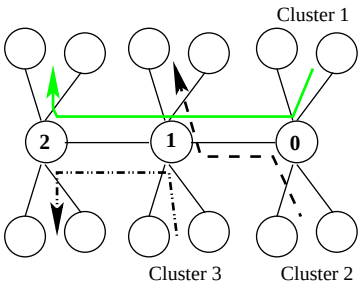

Figure 7: Topology with multiple bottlenecks

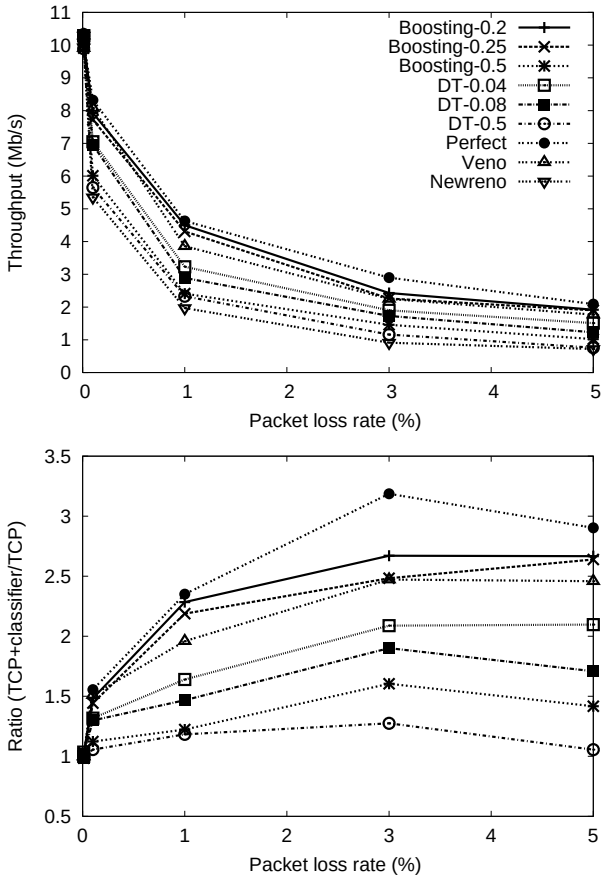

Figure 8: Throughput vs loss rate over a wireless link at the top. The gain (ratio between the throughput of the TCP+classifier and the normal TCP) at the bottom

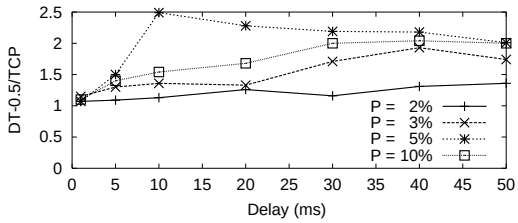

Figure 9: The gain with DT-0.5 versus propagation delay

The classifiers, in this case, have no effect and the gain is not significant.

6.1.1 The gain in very lossy networks (Ad-hoc)

Due to the intermittent connections in ad-hoc networks, TCP flows obtain very low throughput with such technology. The losses are often detected at the expiration of timeouts and are rarely detected by triple duplicate acknowledgments. We thus expect that our classifiers would not perform well.

For this experiment, we used 20 ad-hoc topologies. Each contains 50 mobiles moving according to the model from [11], often used in the literature, and using AODV as the routing protocol. 100 TCP flows have been run (for at least 145 seconds) and the results are recorded in Table 3. The first four columns are respectively the average, maximum, minimum and standard deviation of the ratio between the goodput obtained with our classifier and the goodput obtained by TCP in the same conditions. The last column is the proportion of scenarios for which this ratio is lower than 1, indicating a lower goodput with our classifier than with TCP.

In average, all classifiers improve performance with respect to TCP. In only very few cases (less than 5%),

Table 3: Results in ad-hoc networks

Classifier	avg	max	min	stdev	$p(r < 1)$
Boosting-0.2	1.58	2.70	0.85	0.46	4/100
Boosting-0.25	1.53	2.53	0.97	0.42	1/100
Boosting-0.5	1.23	1.74	1	0.32	0/100
DT-0.04	1.38	2.08	0.87	0.385	3/100
DT-0.08	1.25	1.67	0.9	0.259	1/100
DT-0.5	1.12	1.4	1	0.178	0/100

the use of our classifiers leads to slightly lower goodput than what Newreno would offer in similar conditions. A deeper analysis of our simulations reveals that the difference in these cases is often due to the misclassification of only one or two congestions. The maximum gain can reach 170% (with Boosting-0.2). However, it appears that this huge gain usually happens in very bad conditions, i.e., when the TCP throughput does not exceed 3kbps. In this case, the correct classification of four or five losses allows to avoid timeout expirations and increases the total throughput by 2 or 3kbps, which is negligible but represents a high relative increase. As observed previously the gain with the classifier is not very high in scenarios where most of the losses are not detected early by triple duplicate acknowledgments.

6.1.2 Round-trip time effect

Another important factor that influences the gain of TCP with classifiers is the round-trip-time. When the RTT increases, TCP needs more time to recover its throughput after a loss, and a classifier which does not decrease its throughput has a big advantage over a normal TCP. However, when the round-trip-time increases too much and the flow loses packets, the throughput decreases significantly and the number of sent packets decreases also. Since the error model is based on the number of packets, the number of lost packets decreases, and then the classifier can recover less often. Thus the gain decreases. The number of losses detected by triple duplicates is an important factor on the gain, and the gain depends on everything that can affect the number of losses (e.g. packet size).

We choose here to show the effect of the round-trip-time on the gain in the case of DT-0.5, i.e. the worst model. To this end, we use a simple wireless link with bandwidth equal to 2Mb/s and we vary the propagation delay from 1ms to 50ms. We compare the throughput of TCP with the one of DT-0.5 with different values of the loss rate, and plot the ratio between them on Figure 9. We can see that even DT-0.5, the worst of our models, can offer a good gain.

6.1.3 A note about MAC retransmissions

The simple uniform error model and the two-states Gilbert-Elliott one have been used as basis to learn our classification models. Because they do not take MAC retransmissions into account, these error models have often been considered as too simplistic and that makes questionable the performance of our models in an environment with retransmissions. To give an idea of how our classification models behave in the presence of retransmissions, we have carried out some preliminary experiments with Qarmas, a radio layers simulator developed by France Telecom over Opnet. This simulator is used to carry out performance studies and is considered as very reliable. To ensure this reliability, the simulator is fed by the real algorithms developed and used by the vendors (e.g. Alcatel, Nokia) as well as the exact parameter values. To stick even more to reality, timers are introduced and are calibrated using real experimental results.

We have chosen UMTS for our experiments, with a RAB of 384Kbps. The number of retransmissions, which is vendor specific, was chosen among 3, 7 and unlimited with the usage of a timeout (whose default value was fixed to 1 second). In these experiments, we noticed, unsurprisingly, a higher rate of misclassifications due to link error losses in comparison with scenarios without MAC retransmissions. However, investigation of these misclassifications has shown that more than 90% of the radio layer losses that were misclassified as congestion losses, were actually caused by RLC buffer overflow. So, although they do not correspond to congestion at the IP layer as usually meant, they are nevertheless caused by congestion at the radio layer and the reduction of the congestion window which is induced by the misclassification is actually the right action to take in this situation.

6.2 TCP-friendliness

6.2.1 Wired network

In the previous section we have tested the different models in wireless link, and we have compared the gain obtained by each one. In this section we compare their fairness towards TCP in the wired case, which is an important criterion that should be fulfilled by the classification model. We evaluate TCP-Friendliness on a topology commonly used for this purpose (e.g [1], [25]), illustrated in Figure 5 with $n = 2$. Each experiment consists of running two concurrent TCPs: a standard one (Newreno) and another one using one of the tested models. We plot in Figure 10 the ratio between the throughput of both TCPs. We can see that the Veno

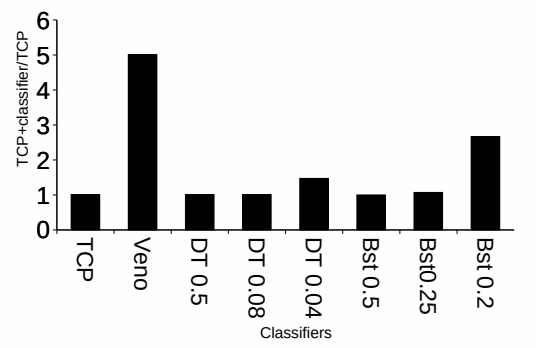

Figure 10: The throughput obtained by each model when they are in competition with normal TCP

classifier is far from being fair. The ratio is close to five which makes it totally unfair towards TCP. Boosting-0.2 offers a ratio slightly lower than three, when DT-0.04 got 1.5 time as high as TCP throughput. All the other models offer a ratio close to one.

6.2.2 Wireless Network

To know if a classifier is TCP-Friendly in a wireless network, we should run two TCPs in competition and replace one of them with a TCP+classifier. If the remaining TCP has similar throughput in both cases, then the TCP classifier is TCP-Friendly. The goal of this section is to show that a protocol can have a higher throughput than TCP in wireless network and still maintain TCP-Friendliness. We carry out this study only with Boosting-0.25 and DT-0.04.

We use the topology of Figure 5 ($n = 2$) where the wireless link is the common one. One flow is TCP and the other is either TCP or TCP with one of the two models. We vary the loss rate of the wireless link from 1 to 9%, and each experiment is run 100 times.

The top of Figure 11 shows the evolution of TCP in the three cases as a function of the loss rate. We can see that the evolution is exactly the same and the curves are very close to each other, which means that the TCP flow gets the same throughput irrespective of whether the competitor is TCP or one of our TCP+classifier. However, Boosting-0.25 and DT-0.04 get much more bandwidth than TCP as we can see on the last two graphs of Figure 11. Thus, a protocol can have more bandwidth than TCP in the wireless case and at the same time maintain TCP-Friendliness. In other words, the additional throughput of Boosting-0.25 and DT-0.04 is not at the expense of a concurrent TCP.

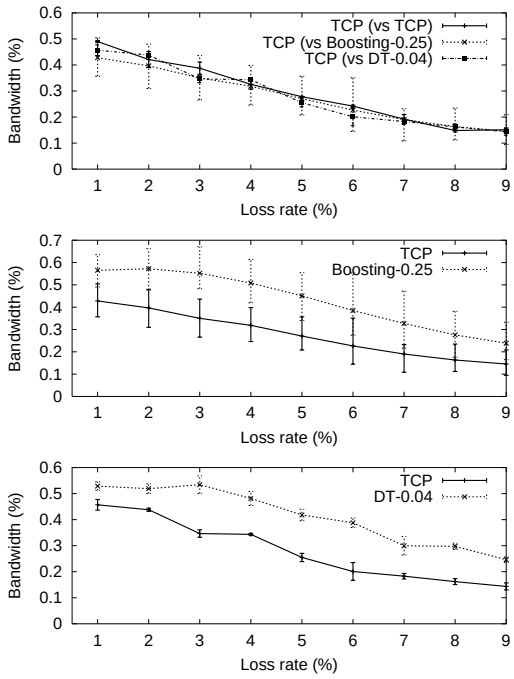

Figure 11: From top to bottom: TCP when competing with TCP Boosting-0.25 and DT-0.04, TCP versus Boosting-0.25, and TCP versus DT-0.04

6.3 Link capacity usage

In section 5.2, we have discussed the effect of misclassifying the congestion on bandwidth usage. To show this effect according to the classification model, we have used an aggregate of 4 similar flows competing over one link (topology of Figure 5 with $n = 4$) and we have computed their goodput, throughput, and efficiency[7]. The results are illustrated in Table 4.

[7]The efficiency is defined as the ratio between the goodput and the throughput.

Table 4: The bandwidth usage

	Throughput	Goodput	Efficiency
NewReno	98.51	96.00	97.45
Veno	94.44	89.68	94.96
DT-0.5	98.51	96.00	97.45
DT-0.08	98.51	96.00	97.45
DT-0.04	99.58	96.04	96.44
Boosting-0.5	98.51	96.00	97.45
Boosting-0.25	99.82	96.77	96.95
Boosting-0.2	100.00	95.75	95.75

On this topology where all losses are obviously due to congestion, Boosting-0.5, DT-0.5, and DT-0.08 perfectly classify all packet loss causes. Consequently, these methods give exactly the same results as TCP NewReno. Veno's rule is the classification method that gives the worst results along the three criteria. Its low bandwidth usage is a consequence of the misclassification of congestion losses that worsen the congestion and finally trigger timeouts. Boosting-0.2 occupies slightly more bandwidth than TCP NewReno, but it provides a lower goodput. Its behaviour is similar to Veno but its Err_C is still lower. DT-0.04 and Boosting-0.25 yield a better link usage than NewReno and also a better goodput. Nevertheless, their efficiency is slightly lower than NewReno's. Some congestion losses are misclassified by these rules, which prevents TCP from reducing its window and yields a more stable throughput. However, unlike with Veno's rule, the misclassifications are not frequent enough to lead to timeout expirations and thus the bandwidth is higher than with NewReno and close to the capacity of the link. On the other hand, DT-0.04 et Boosting-0.25 lose more packets and thus require more retransmissions, which explains their lower efficiency.

6.4 Multiple bottlenecks

Until now, all the experiments have been carried out over topologies with one bottleneck. In this section, for comparison, we use the topology with two bottlenecks illustrated in Figure 7. The bandwidth of both bottlenecks is chosen at 3Mb/s. In each cluster, we have one Newreno and one of our classifiers. We compute the throughput and the goodput of each flow and summarize them in Tables 5, 6, and 7. Note that DT-0.5 and Boosting-0.5 , which made no classification errors during this experiment, behave exactly like TCP NewReno. So, in all tables, the line corresponding to NewReno and that of DT-0.5 and Boosting-0.5 are similar.

Tables 5 and 6 show that Boosting-0.2 consumes less bandwidth and offers the lowest efficiency. Again, this is a consequence of its more frequent misclassifications of congestion losses that trigger timeout expirations. The other classifiers are quite equivalent. From these three

Other classifiers provide a better usage of the resources. As they better classify loss causes, they often reduce their rate together with the TCP flow belonging to the same cluster and this yields a better usage of the links without affecting fairness. Table 7 shows the ratio between the throughput of TCP and the throughput of the other flow in the same cluster. We observe that the

Table 5: Efficiency of each cluster (%)

	Cluster 1	Cluster 2	Cluster 3
NewReno	95.36	98.93	98.87
DT-0.5	95.36	98.93	98.87
DT-0.08	97.95	98.59	99.01
DT-0.04	98.55	98.69	98.74
Boosting-0.5	95.36	98.93	98.87
Boosting-0.25	94.76	99.01	99.07
Boosting-0.2	95.41	98.16	98.45

Table 6: Throughput, goodput, and efficiency (%) in paths 0-1 (top) and 1-2 (bottom)

	Throughput	Goodput	Efficiency
NewReno	90.67	89.00	98.16
DT-0.5	90.67	89.00	98.16
DT-0.08	94.55	92.6	97.94
DT-0.04	95.63	93.72	98.01
Boosting-0.5	90.67	89.00	98.16
Boosting-0.25	97.33	95.67	98.29
Boosting-0.2	93.56	91.09	97.36

	Throughput	Goodput	Efficiency
NewReno	96.00	94.33	98.26
DT-0.5	96.00	94.33	98.26
DT-0.08	96.21	94.56	98.28
DT-0.04	95.61	93.75	98.05
Boosting-0.5	96.00	94.33	98.26
Boosting-0.25	98.00	96.33	98.30
Boosting-0.2	94.75	92.45	97.58

Table 7: Fairness ratio

	Clust 1	Clust 2	Clust 3	Average
NewReno	1.59	1.08	1.12	1.26
DT-0.5	1.59	1.08	1.12	1.26
DT-0.08	1.33	1.05	1.08	1.15
DT-0.04	1.16	1.29	1.01	1.15
Boosting-0.5	1.59	1.08	1.12	1.26
Boosting-0.25	1.09	1.16	1.16	1.14
Boosting-0.2	1.22	1.08	1.28	1.19

Table 8: Throughput (%) in a dynamic scenario

	0-60s	60-100s	100-140s	140-200s
NewReno	62.99	64.79	62.87	63.51
Dt-0.04	91.50	73.46	94.73	72.63

ratios are very close to the one provided by NewReno. So, all tested models offer good fairness towards the TCP flow belonging to their cluster.

6.4.1 Dynamic scenario

Since our classifiers are based on past measurements, one could imagine that their accuracy and efficiency could decrease in dynamic scenarios where the load changes throughout the same connection from high to low or from low to high. However, the memory of our classifiers does not exceed two round-trip times. Therefore, the values of the inputs are only helpless for classifying losses during the two round-trip times following the load change. The worst case is thus when a loss occurs in each of the two RTTs following the load changes and it can only lead to two misclassifications. We thus expect that load changes will essentially have a negligible impact on our results.

The multiple bottleneck topology of Figure 7 has been used to show that the accuracy and the efficiency are indeed maintained in dynamic scenarios. The link 1-2 has been replaced by a wireless link and now each cluster does not contain more than one flow. The simulation consists in running cluster 1 alone for 40 seconds, then starting cluster 2 and stopping it after 60 seconds and finally running cluster 3 after 140 seconds. Table 8 compares the average throughput of NewReno and DT-0.04 over the different time intervals. The DT-0.04 classifier still clearly outperforms TCP. As another control experiment, we ran the four simulations independently (rather than in sequence), so as to eliminate any bias due to past events, and we observed similar results.

6.5 Discussion

Figure 12 summarizes the results obtained in the previous sections. It illustrates the gain versus the fairness for all the models we have tested in the previous scenarios. The vertical line represents the limit of TCP-friendliness according to [17], i.e. $x = 1/1.78 = 0.56$. Classifiers on the right of this line are considered to be TCP-friendly and the closer a classifier is to the line $x = 1$, the fairer it is towards TCP.

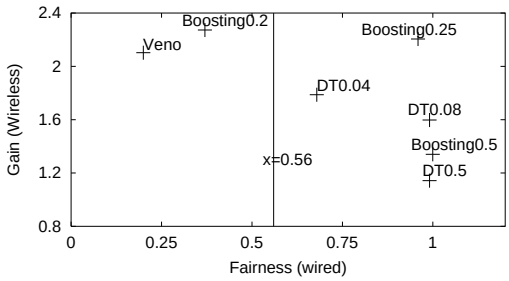

Figure 12: Fairness (over wired links) versus the gain (over wireless links)

If we forget the computational constraints, the Pareto optimal classifiers are Boosting-0.2, Boosting-0.25, DT-0.08, and Boosting-0.5. Boosting-0.2 gives the best gain in wireless link but it is not TCP-friendly. Among the other classifiers, Boosting-0.25 is the one that clearly provides the best tradeoff between gain and fairness. Its gain is only slightly lower than the gain of Boosting-0.2 and it is only slightly less fair than DT-0.08 and Boosting-0.5. Furthermore, our experiments in Section 6.3 have shown that it correctly uses the link capacity. So, we advocate its use in all situations.

Nevertheless, for computational efficiency reasons, we may prefer to use single decision trees that compute faster classification and require less memory. In this case, the two classifiers of interest are DT-0.04 and DT-0.08. DT-0.08 offers a very good fairness and a reasonable gain. DT-0.04 is less fair but it has a higher gain. Strictly, DT-0.04 can still be considered as TCP-friendly because it lies at the right of the line $x = 0.56$. However, its misclassification rate on congestion is equal to about 29% which is much higher than the maximum value of Err_C which was computed in Section 5.3. Hence, if TCP-friendliness is very important, DT-0.08 should be preferred.

7 Conclusion

In this paper, we have proposed to apply supervised learning techniques to automatically derive a classifier that can distinguish losses due to congestion from losses due to link errors in the case of wired/wireless networks. To this end, we have gathered an important number of losses together with their causes by simulating different random topologies and we have applied on the resulting learning sample different supervised learning algorithms, namely Decision trees, Tree bagging, Tree boosting, and Neural networks. Each method gives a different model and, by adjusting one common parameter, can offer different tradeoffs between the two

types of error: error on losses due to congestion versus error on losses due to link errors. All methods gave quite good results in terms of error rates and, furthermore, they are all much better than the simple ad-hoc classification rules used in other protocols, like Veno or Westwood. Among all methods, the tree boosting algorithm gives the best results.

Then, we have proposed to extend TCP with such a classifier by simply preventing TCP from reducing its window size when the loss is classified as due to a link error. This protocol was evaluated from the point of view of throughput and TCP-friendliness in wired and wireless networks.

From an analytical derivation, we have shown that to be TCP-friendly, the classification model should misclassify less than 18% of the losses due to congestion. Our simulations have shown that, when the model is tuned to satisfy this constraint, it is possible to increase significantly the throughput over wireless links while maintaining TCP-friendliness in all cases.

Among all methods, tree boosting offers the most significant gain. Decision trees are an interesting alternative as they still provide a significant gain but at a lower computational cost.

In this paper, we have tested the efficiency of our classification models in a very straightforward extension of Newreno. Nevertheless, it is clear that we could also combine them with more sophisticated protocols that have been proposed for wireless networks like for example Veno or Westwood. We believe that this could greatly improve these protocols given the very poor quality of the classification rules they use.

We see three potential limitations to our approach. First, we have not taken into account losses detected by timeout expiration, which were thus all considered as signs of congestion. However, even without classifying such losses, the throughput gains observed in our simulations are already excellent and we can only improve these results by taking into account timeout expirations when designing the loss classifier.

A second potential limitation is that our database was generated from simulated networks (topologies and traffics) which may differ to a certain extent from actual ones. However, the learning sample was generated by randomizing all networks conditions. Hence, there is no bias in the classifier. We have also carried out experiments with a more realistic topology generator (BRITE [29]) that have not shown significant differences in terms of performance of the classifiers with respect to the results presented in this paper. Furthermore, restraining the learning to actual topologies and flows can only improve the accuracy of the already excellent classification.

Another potential limitation is that the performance of our classification models could depend on the choice of the queuing policies and the error models. Actually, we have carried out experiments with several combinations of queuing policies (Droptail or RED) and error models (uniform or Markovian-Gilbert model) to see if these parameters influence the resulting classification model. It appears from these experiments that, for a given queuing policy, the error model does not influence the classification model. On the other hand, it is more difficult to classify losses with RED than with Droptail. This is not surprising since congestion losses with Droptail are due to a buffer overflow while they can be more random in RED. Nevertheless, loss classifications when RED is used are not so bad. For example, the error rate with DT-0.5 goes from 6.34% with Droptail to 9.37% with RED.

8 Acknowledgments

The authors would like to thank Olivier Bonaventure for his helpful comments on an earlier version of this paper. This work has been partially supported by the Belgian Science Policy in the framework of the IAP programme (MOTION P5/11 project), by the E-NEXT European Network of Excellence (NoE) and by the European IST-FET ANA project. P.Geurts is a Research Associate at the National Fund for Scientific Research (FNRS, Belgium).

References

[1] E. Altman, C. Barakat, and V. M. Ramos. Analysis of AIMD protocols over paths with variable delay. In *Proceedings of IEEE INFOCOM*, March 2004.

[2] A. Bakre and B. R. Badrinath. I-tcp: indirect tcp for mobile hosts. In *Proceedings of the 15th Int. Conf. on Distributed Computing Systems*, 1995.

[3] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 256–269. ACM Press, 1996.

[4] D. Barman and I. Matta. Effectiveness of loss labeling in improving tcp performance in wired/wireless networks. In *Proceedings of the 10th IEEE International Conference on Network*

Protocols, pages 2–11. IEEE Computer Society, 2002.

[5] D. Barman and I. Matta. Model-based loss inference by tcp over heterogeneous networks. In *In Proceedings of WiOpt 2004*, pages 364–373, March 2004.

[6] S. Biaz and N. H. Vaidya. Distinguishing Congestion Losses from Wireless Transmission Losses: A Negative Result. In *Proceedings of IC3N, New Orleans*, 1998.

[7] S. Biaz and N. H. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. In *Proceedings of the 1999 IEEE Symposium on Application - Specific Systems and Software Engineering and Technology*, page 10. IEEE Computer Society, 1999.

[8] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[9] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[10] L. Breiman, J. Friedman, R. Olsen, and C. Stone. *Classification and Regression Trees*. Wadsworth International (California), 1984.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM Press.

[12] S. Cen, P. C. Cosman, and G. M. Voelker. End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Transactions on Networking*, 11(5):703–717, Oct. 2003.

[13] R. Duda, P. Hart, and S. D.G. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000.

[14] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, HP Labs, 2004.

[15] S. Floyd. TCP and Explicit Congestion Notification. *ACM Computer Communication Review*, 24:10–23, October 1994.

[16] S. Floyd. A report on some recent developments in TCP congestion control. *IEEE Communication Magazine*, 39:84–90, April 2001.

[17] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of SIGCOMM 2000*, pages 43–56, 2000.

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2th European Conference on Computational Learning Theory*, pages 23–27, 1995.

[19] C. P. Fu and S. C. Liew. TCP Veno: TCP enhancement for transmission over wireless access networks. *IEEE JSAC*, Feb. 2003.

[20] P. Geurts, I. El Khayat, and G. Leduc. A machine learning approach to improve congestion control over wireless computer networks. In *Proceedings of IEEE Int. Conf. on Data Mining (ICDM-2004)*, pages 383–386, 2004.

[21] A. Gurtov and S. Floyd. Modeling wireless links for transport protocols. *SIGCOMM Computer Communication Review*, 34(2):85–96, 2004.

[22] M. Handley, J. Padhye, and S. Floyd. TCP friendly rate control (TFRC): Protocol specification. *RFC 3448, Proposed Standard*, January 2003.

[23] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2001.

[24] I. E. Khayat, P. Geurts, and G. Leduc. http://www.run.montefiore.ulg.ac.be/~elkhayat/Boosting-DT/Boosting-dt.html.

[25] K.-W. Lee, R. Puri, T. eun Kim, K. Ramchandran, and V. Bharghavan. An Integrated Source Coding and Congestion Control Framework for Video Streaming in the Internet. In *INFOCOM (2)*, pages 747–756, 2000.

[26] J. Liu, I. Matta, and M. Crovella. End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[27] M. Mathis, J. Semke, Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 3:67–82, July 1997.

[28] S. McCanne and S. Floyd. *The LBNL Network Simulator*. Lawrence Berkeley Laboratory, `http://www-nrg.ee.lbl.gov/ns/`, 1997.

[29] A. Medina, I. Matta, and J. Byers. BRITE: A Flexible Generator of Internet Topologies. Technical report, Boston University, 2000.

[30] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, 2000.

[31] C. Parsa and J. J. Garcia-Luna-Aceves. Improving tcp congestion control over internets with heterogeneous transmission media. In *Proceedings of the Seventh Annual International Conference on Network Protocols*, page 213. IEEE Computer Society, 1999.

[32] C. Parsa and J. J. Garcia-Luna-Aceves. Differentiating congestion vs. random loss: a method for improving TCP performance over wireless links. In *Proceedings of IEEE WCNC'2000*, pages 90–93, 2000.

[33] C. Parsa and J. J. Garcia-Luna-Aceves. Improving TCP performance over wireless networks at the link layer. *Mob. Netw. Appl.*, 5(1):57–71, 2000.

[34] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda. Achieving moderate fairness for udp flows by path-status classification. In *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*, page 252. IEEE Computer Society, 2000.

[35] V. Tsaoussidis and H. Badr. TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains. In *Proceedings of the 8th International Conference on Network Protocols (ICNP)*. IEEE Computer Society, 2000.

[36] V. Tsaoussidis and I. Matta. Open Issues on TCP for Mobile Computing. *J. Wireless Commun. Mobile Comput.*, 2002.

[37] V. Tsaoussidis and C. Zhang. The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks. *IEEE Journal on Selected Areas in Communications*, 23(6):1178–1189, June 2005.

[38] R. Wang, M. Valla, M. Sanadidi, B. Ng, and M. Gerla. Efficiency/Friendliness Tradeoffs in TCP Westwood. In *Proceedings of the 7th IEEE Symposium on Computers and Communications*, 2002.

[39] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma. Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement, 2002.

[40] K. Xu, Y. Tian, and N. Ansari. TCP-Jersey for Wireless IP Communications. *IEEE Journal on Selected Areas in Communications*, 22:747–756, May 2004.

[41] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *Communications Magazine, IEEE*, 39(4):52–58, 2001.