Robust Regression for Minimum-Delay Load-Balancing

Federico Larroca and Jean-Louis Rougier
Télécom ParisTech, Paris, France - Email: name.surname@telecom-paristech.fr

***Abstract*—By providing origin-destination pairs with several possible paths, *Dynamic Load-Balancing* has been shown to obtain excellent results in terms of robustness and effective resource usage. In these dynamic schemes, paths are defined a priori, and the portion of traffic routed through each path is (typically) adjusted so that the sum over all links of a certain link-cost function is minimized. Queueing delay is usually used as this cost function due to its versatility and simplicity. However, all load-balancing schemes require an analytical expression of the delay, for which oversimplistic models are used (such as the classic *M/M/1* model). In this paper we propose a framework that instead learns this queueing delay function from measurements, while restricting the assumptions to the minimum. For this, we use a novel robust regression method that, given a set of link load and delay measurements, returns a very simple regression delay function. Some adjustments to this regression function allow us to use it as the link cost of a greedy load-balancing algorithm that converges to the actual minimum-delay configuration. We also compare our framework with previous load-balancing proposals, showing for instance that using the *M/M/1* model results in a total delay that may easily exceed the minimum by 10%, and can go as high as more than 100%.**

I. Introduction

Network convergence is a reality. Many new services such as P2P or HD-TV are offered on the same network, increasing the unpredictability of traffic patterns. To make matters worse, access rates have increased at such pace that the old assumption that core link capacities are several orders of magnitude bigger than access rates is no longer true. Moreover, there are new emerging architectures with intrinsically scarce resources (e.g. Wireless Mesh Networks). Thus, simply upgrading link capacities may not be a viable solution any longer. This means that network operators are now, more than ever, in need of Traffic Engineering (TE) mechanism which are efficient (make good use of resources), but also automated (as much self-configured as possible), more robust with respect to network variations (changes in traffic matrix, or characteristics of transported flows) and more tolerant (in case of node/link failures).

Dynamic load-balancing (DLB) [1]–[3] is a TE mechanism that meets these requirements. If an origin-destination (OD) pair is connected by several paths, the problem is simply how to distribute its traffic among these paths in order to achieve a certain objective. In these dynamic schemes, paths are configured a priori and the portion of traffic routed through each of them (demand vector) depends on the current traffic matrix (TM) and network condition. Since DLB only uses the present situation as an input, it clearly obtains the best performance out of the available resources. For instance, Robust Routing [4] (which finds a single routing configuration to support a whole set of TMs) can only guarantee a worst-case performance and support a subset of the TMs that DLB can support.

Formally, DLB is generally defined in terms of a link-cost function of the load, where the demand vector is adjusted in order to minimize the sum over all links of their respective cost. The rationale behind this definition is that the cost function should represent the congestion on the link, for which the queueing delay is generally used. The choice is justified by its versatility (big queueing delays mean bad performance for all traffic) and simple algebra (the total delay of a path is the addition over its links of their respective delay). However, most DLB schemes require an analytical formula of this delay, for which classic and oversimplistic models (e.g. *M/M/1* [5]) are used [1], resulting in a not so good performance [6] and, as we shall show, an actual total delay that is significantly bigger than the optimum.

In this paper, we propose a framework that makes no assumption on the delay function, other than some reasonable hypothesis on its shape (e.g. delay may not decrease with load). We learn this function from measurements instead, for which we turn our attention to the recent work of Kuosmanen [7]. Given a set of measurements of link load and delay, the method allows us to obtain a very simple regression (or approximative) function that fulfills the required shape constraints. This function can be easily adapted to be used as a link-cost function by a greedy load-balancing algorithm that converges to the actual minimum-delay configuration.

In an initial version of our framework [8], we concentrated on analyzing what constitutes a "good" set of measurements and discussed some shortcomings of the resulting regression function. In this paper we will first verify that the homoscedasticity assumption (i.e. measurement errors have all the same variance) made in [7] (and thus in [8]) is not suitable in our context. This heterogeneous variance, together with the presence of outliers, may have an important negative impact on the resulting regression. We will then derive a similar but more robust regression method, and compare its performance with the original one. Furthermore, we will extend the study by presenting simulations with a real topology and several real

This work was partially funded by CELTIC project TRANS and FP7 project Euro-NF

TMs, which indicate that the gain achieved by our framework in terms of delay over an oversimplistic approximation (like the *M/M/1*) may be significant. Moreover, the comparison with previous proposals in terms of link utilization shows that our framework either outperforms them, or the difference is not significant.

The rest of the paper is structured as follows. In the following section we define the network model and discuss the greedy algorithm used to minimize the total delay when the link delay function is known. In Sec. III we present the method to obtain a robust approximation of the delay function from past measurements, and derive some necessary adjustments to the approximate function to enable its use with the greedy algorithm. We make a performance analysis of our framework in Sec. IV, where we compare it with previous load-balancing methods and we discuss some implementation issues. We conclude the paper in Sec. V.

II. Greedy Load-Balancing

A. Network Model

The network is defined as a graph $G = (V, E)$. In it there are a number of so-called *commodities* (or OD pairs), indexed by $s = 1, .., S$, specified in terms of the triplet o_s, q_s and d_s; i.e. origin node, destination node and a certain fixed demand of traffic from the former to the latter. Each commodity s can use n_s paths connecting o_s to q_s (each noted as P_{si} for $i = 1, .., n_s$), and can distribute its total demand arbitrarily among them. Commodity s sends an amount d_{si} of its traffic through path P_{si}, where $d_{si} \geq 0$ and $\sum d_{si} = d_s$. This distribution of traffic induces the demand vector $d = (d_{si})$.

Given the demand vector, the total load on link l is then $\rho_l = \sum_s \sum_{i:l \in P_{si}} d_{si}$. The presence of this traffic on the link induces a certain mean queueing delay given by the non-decreasing function $D_l(\rho_l)$. The total delay of path P is defined as $D_P = \sum_{l:l \in P} D_l(\rho_l)$. As a measure of the congestion in the network, we shall use the *mean total delay* $D(d)$ defined as:

$$D(d) = \sum_{s=1}^{S} \sum_{i=1}^{n_s} d_{si} D_{P_{si}} = \sum_{l=1}^{L} D_l(\rho_l)\rho_l := \sum_{l=1}^{L} f_l(\rho_l)$$

That is to say, a weighted mean delay, where the weight for each path is how much traffic is sent through it, or in terms of the links, the weight of each link is how much traffic is traversing it. We prefer this congestion measure to a simple total delay because it reflects more precisely performance as perceived by traffic. Two situations where the total delay is the same, but in one of them most of the traffic is traversing heavily delayed links should not be considered as equivalent. Note that, by Little's law, $f_l(\rho_l) := D_l(\rho_l)\rho_l$ is proportional to the average number of bytes in the queue of link l. We will then use this last value as $f_l(\rho_l)$ which (like the mean load) is readily available in most routers.

We are now in conditions to write the problem explicitly:

$$\underset{d}{\text{minimize}} \sum_{l=1}^{L} f_l(\rho_l) \qquad \text{s.t.} \quad d_{si} \geq 0 \ \sum_{i=1}^{n_s} d_{si} = d_s \qquad (1)$$

Note that no explicit constraint on ρ_l was made. This is assumed to be implicitly included in the delay function. For instance, $f_l(\rho_l)$ goes to infinity (or a relatively high value) as ρ_l reaches c_l (the link capacity) and remains at infinity after this point. It should also be noted that in the framework described above the destination for a commodity is not necessarily a single node (e.g. two gateways to the internet may be seen as a single destination).

B. Wardrop Equilibrium

In this section we present and discuss how to solve problem (1) in a distributed fashion. In particular, we will consider mechanisms where each commodity greedily minimizes a certain cost function of its paths (ϕ_P), which require minimum coordination. This context constitutes an ideal case study for game theory, and is known as *Routing Game* in its lingo [9]. The case in which the path cost is the sum over its links of a positive non-decreasing link-cost function of the load ($\phi_P = \sum_{l:l \in P} \phi_l(\rho_l)$) is known as *Congestion Routing Game* and has several important properties such as uniqueness of the equilibrium.

In a routing game like ours, commodities are assumed to be constituted by infinitely many agents, each controlling an infinitesimal amount of the demand. Each of these agents (or players) decides through which path to send its traffic. In this context the division d_{si}/d_s represents the portion of agents of commodity s that have P_{si} as their choice. If every agent acts selfishly, then the system will be at equilibrium when no agent can decrease its cost by changing its path choice. This situation constitutes what is known as a *Wardrop Equilibrium* (WE) [10], which is defined as follows.

Definition 1: A demand vector is a Wardrop Equilibrium if for each commodity $s = 1 \ldots S$ and for each path P_{si} with $d_{si} > 0$ it holds that $\phi_{P_{si}} \leq \phi_{P_{sj}}$ for all P_{sj} with $j = 1, .., n_s$.

It may be proved that a WE results in a local minimum of the so-called potential function [9]:

$$\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \phi_l(x)dx$$

This means that, given an objective function like (1), we may find a cost function such that the resulting WE is the optimum demand vector. In this case, let us consider the following cost function:

$$\phi_l(\rho_l) = \frac{\partial f_l(\rho_l)}{\partial \rho_l} \Rightarrow \qquad (2)$$

$$\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \frac{\partial f_l(x)}{\partial x} dx = \sum_{l=1}^{L} (f_l(\rho_l) - f_l(0))$$

This means that the Wardrop Equilibrium of a congestion routing game where the link cost is the derivative of $f_l(\rho_l)$

results in a local minimum of (1) (note that since $f_l(0)$ is a constant, the optimum d is the same with or without its addition). Moreover, if $f_l(\rho_l)$ is convex, this local minimum is then the unique global minimum demand vector. A distributed algorithm that converges towards such equilibrium is described in the following subsection.

C. REPLEX: Exploration-Replication Policy

The concept of Wardrop Equilibrium was first proposed in the context of transportation to characterize the equilibrium of users who greedily want to minimize their travel time. In this context, users are assumed rational and their behavior is the mechanism through which the equilibrium is attained. However, in our case routers make the choice for every user (i.e. packets), and an algorithm that when independently ran in every router reaches the equilibrium as fast as possible and does not oscillate has to be specified. In [11] the authors present such mechanism and use it to design a load-balancing scheme called REPLEX in [3], which we now briefly describe.

At regular intervals, each OD pair changes a portion of traffic from the paths with bigger total cost ϕ_P to those with a smaller one. The exact amount is proportional to the relative difference in cost times a parameter that controls the algorithm's speed (noted as λ). Details on the algorithm may be consulted on the references, although we shall further highlight the fact that convergence is guaranteed as long as λ is smaller than k/r, where k is a suitable constant and r is an upper-bound to the relative slope of all $\phi_l(\rho_l)$, which is defined as follows:

Definition 2: A differentiable cost function $\phi_l(x)$ has *relative slope* r at x if $\phi_l'(x) \leq r\phi_l(x)/x$. A cost function has relative slope r if it has relative slope r over the entire range $[0, 1]$.

Intuitively, migration from one path to the other should be slow if the cost function has abrupt changes. On the other hand, if the cost function is relatively "soft", changes may be faster.

III. Non-Parametric Regression With Shape Restrictions

A. Weighted Convex Non-Parametric Least Squares

The problem we address now is how to learn $f_l(\rho_l)$ from measurements (actually we are interested in its derivative, $\phi_l(\rho_l)$, but we can only observe the queue size $f_l(\rho_l)$). For the sake of clarity we will concentrate on the problem for a single link, so we shall omit the sub-index l. We are given n pairs of observations $(\rho_1, Y_1), (\rho_2, Y_2), \ldots, (\rho_n, Y_n)$ (also called *training set*), where the *response variable* Y (the measured mean queue size) is related to the *covariate* ρ (the link load) by the equation:

$$Y_i = f(\rho_i) + \epsilon_i \qquad i = 1, \ldots, n \tag{3}$$

Where $f(\rho)$ is now called the *regression function* and the measurement errors $\epsilon = (\epsilon_1, \ldots, \epsilon_n)'$ are assumed to be uncorrelated random variables with $\mathbb{E}(\epsilon_i) = 0$ and $\mathrm{Var}(\epsilon_i) = \sigma_i^2 < \infty$. The problem is to "learn" $f(\rho)$ from the observations in the training set and obtain an estimation $\widehat{f}(\rho)$, restricting the assumptions on its functional form as much as possible. So far, we have only these three necessary requirements:

1) $f(\rho)$ is clearly non-decreasing (more load may never lead to a smaller queue size).
2) $f(\rho)$ should be convex. This is to guarantee the existence and uniqueness of the optimum, and its coincidence with the WE.
3) $\phi(\rho)$ should have a finite relative slope in order to enable a correct operation of REPLEX (and probably all distributed optimization algorithms).

We will now consider the two first requirements, which are by far the most restrictive. There are several regression methods that make *no* assumptions on the regression function and allow one to obtain its derivative (for a good overview of this and other regression methods see [12]). For instance, *Local Polynomial Regression* is a kernel-type regression method that allows one to estimate any order derivative of the regression function through a standard weighted least square. However, it presents several problems. First of all, estimating the function at any point is as costly as learning it, i.e. a weighted least square problem has to be solved every time the function or its derivative wants to be estimated. In a way, the functional representation of $\widehat{f}(\rho)$ is the whole training set, which can be relatively big. Secondly, all kernel-type methods suffer from the so-called bias-variance tradeoff, controlled by the bandwidth parameter, which can be very tricky to assign and on which the quality of the estimation depends heavily. Finally, in order to enforce shape constraints, such as monotonicity and convexity, "indirect" methods have to be used. For instance, [13] describes a method to transform the training set so that when the local polynomial method is applied, shape restrictions are assured. Anyway, the intrinsic problems of kernel-type methods we already mentioned are still present.

We turn our attention then to the *Convex Nonparametric Least Squares* (CNLS) problem [7]. Let $\mathcal{F}$ be the set of continuous, monotonic increasing and globally convex functions. CNLS consists on finding $\widehat{f} \in \mathcal{F}$ that minimizes the sum of squares of the residuals. However, this minimization is known to be very sensitive to outliers, and results in a biased estimator when measurement errors present heteroscedasticity. As we shall see in the examples, both kinds of problems are present in our measurements. A possible solution to these issues is to consider a weighted version of the original problem, which we shall call *Weighted Convex Nonparametric Least Squares* (WCNLS):

$$\min_f \sum_{i=1}^{n} \omega_i \left(Y_i - f(\rho_i)\right)^2 \quad \text{s.t. } f \in \mathcal{F} \tag{4}$$

where the weight ω_i is a positive constant that indicates the importance of each observation. For instance, outliers should have a small weight. Due to the size of $\mathcal{F}$, problem (4) is very difficult to solve in such general version. The idea is to identify a subset of representor functions $\mathcal{G} : \mathcal{G} \subset \mathcal{F}$ such that by substituting the constraint $f \in \mathcal{F}$ by $f \in \mathcal{G}$ the

resulting optimum is left unchanged but the problem is easier to solve. Consider then the following family of piecewise linear functions:

$$\mathcal{G}(P) = \Big\{ g : \mathbb{R} \to \mathbb{R} \mid g(\rho) = \max_{i=1,\dots,n} \alpha_i + \beta_i \rho; \\ \beta_i \geq 0 \ \ \forall i = 1, \dots, n; \\ \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \ \ \forall j, i = 1, \dots, n \Big\}$$

It is clear that $\mathcal{G}(P)$ belongs to $\mathcal{F}$ for any arbitrary set of observations $P = \{\rho_i\}_i$. It turns out that we may substitute $\mathcal{G}(P)$ in (4) and obtain the same optimal solution. This equivalence may be demonstrated just like in the original CNLS [7]. Its demonstration relied on the fact that the optimization problem depends only on the value of $\widehat{f}(\rho)$ at a finite set of points ρ_i, which is also the case for WCNLS. This result allows us to transform the infinite dimensional problem (4) into the following standard finite dimensional Quadratic Programming (QP) problem:

$$\min_{\epsilon,\alpha,\beta} \sum_{i=1}^{n} \omega_i \epsilon_i^2 \qquad (5)$$

$$\text{subject to} \quad Y_i = \alpha_i + \beta_i \rho_i + \epsilon_i \ \forall i = 1, \dots, n$$
$$\alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \ \forall j, i = 1, \dots, n$$
$$\beta_i \geq 0 \ \forall i = 1, \dots, n$$

Regarding the set of representor functions $\mathcal{G}(P)$, it may seem that we actually transformed a nonparametric problem into a parametric one. However, it should be noted that although we look for a piecewise linear function, the partition of the linear segments is not fixed a priori. That is to say, the number and location of the segments are endogenously determined to minimize the weighted squared residual. Moreover, although problem (5) is a standard QP problem for which mature methods to solve it exist (such as interior point algorithms) and that several solver software are available (for instance, we used MOSEK [14]), its size is considerable. It has a total of $n + n + n$ variables and $n + n(n-1) + n$ constraints. The second set of constraints, which are the key to enforcing the convexity of $\widehat{f}(\rho)$, are quadratic in the number of observations. The size of the problem is clearly the major drawback of the method.

We will now discuss how to assign the values of ω_i. A typical procedure to address both heteroscedasticity and outliers in the classic least squares problem is to iteratively change the weights, in what is known as Iteratively Reweighted Least Squares. This kind of methods begins with arbitrary weights, perform the regression (the result at iteration t is called $\widehat{f}_t(\rho)$), and re-calculate the weights depending on the distance between the measurements and the regression (noted as ω_i^t). The last two steps are repeated until the weights converge. For instance, if the weights are recalculated as $\omega_i^{t+1} = 1/|\widehat{f}_t(\rho_i) - Y_i|$, in the limit the resulting weighted least squares problem finds the minimum of the sum of absolute errors, which is known to be very robust to outliers. As mentioned earlier, the size of problem (5) will represent a problem if we were to solve it several times. We shall then proceed as follows. We perform an initial simple estimation $\widehat{f}_0(\rho_i)$, and calculate the weights as:

$$\omega_i = \frac{1}{\left| \widehat{f}_0(\rho_i) - Y_i \right|} \qquad (6)$$

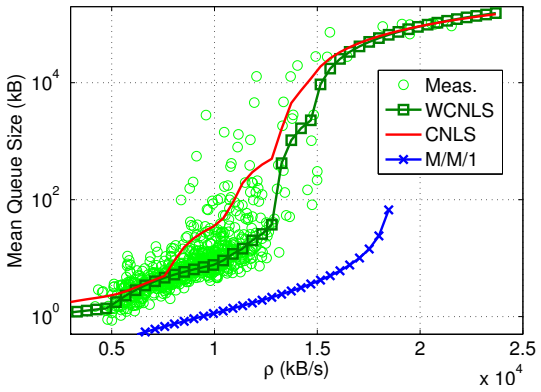
As the initial $\widehat{f}_0(\rho_i)$ we used the *k-nearest neighbors* algorithm (with $k = 10$), which simply estimates $f(\rho)$ as the median of the k measurements Y_i corresponding to the ρ_i's nearest to ρ. In this way, we seek to find a curve that fits the bulk of the data, and minimize the effect of outliers.

B. An Example

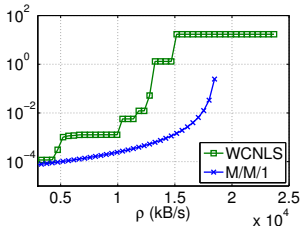
To illustrate the method we will apply it to a training set obtained by injecting a 72 hours long packet trace (obtained from [15]) to a simple queue emulator we developed. In the absence of information we assumed a relatively big buffer size of 100 MB. Measurements correspond to the mean load and queue size in a 60 seconds period, obtained from 12 hours spanning the complete second day of the original trace. Figure 1(a) shows the 720 measurements, the estimated function $\widehat{f}(\rho)$ through WCNLS and CNLS, and the estimation the *M/M/1* model yields $(\rho/(c-\rho))$, all in logarithmic scale for the sake of clarity. First of all, it should be noted that the *M/M/1* model has little to do with the measurements. It consistently underestimates them, except at light loads, where they are relatively small. Regarding the estimation by CNLS, we can see that it very similar to WCNLS at light loads, where the variance is relatively constant and there are almost no outliers. However, after $\rho \approx 8000$ kB/s, several notorious outliers start to appear. Note how CNLS is very influenced by these outliers and how the resulting estimation does not represent the majority of the measurements. On the other hand, WCNLS ignores these outliers almost completely and represents the mean queue size more accurately. Finally, it should be noted that measurements verify the convexity condition, meaning that this requirement is not only required mathematically, but is also verified by reality.

In Fig. 1(b) we show the estimation of the derivative $\phi(\rho)$ through WCNLS and the *M/M/1* model $(c/(c-\rho)^2)$. Since the WCNLS estimation is piecewise linear, the estimation of $\phi(\rho)$ is a piecewise constant function, and outside the support of the observation it becomes constant. As a consequence, WCNLS will produce a good estimation of $\phi(\rho)$ inside this region, after which it will systematically underestimate it. The *M/M/1* model again underestimates the derivative throughout the range of observations, except at light loads where they are both small.

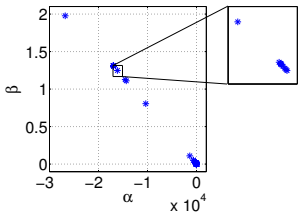
Finally, Fig. 1(c) shows the pairs (α_i, β_i) in the plane. Notice how, although there are 720 different values, and as in the original CNLS, significantly different pairs are relatively few. In all the regressions we performed, the actual number of different pairs resulted in a very small fraction of n. This means that, in contrast to kernel-type regressors, $\widehat{f}(\rho)$



(a) Measurements, $\widehat{f}(\rho)$ for CNLS and WCNLS, and the *M/M/1* model



(b) $\widehat{\phi}(\rho)$ for WCNLS, and the *M/M/1* model



(c) The pairs (β_i, α_i) in the plane

Fig. 1. An example of a regression

is completely represented by a number of parameters that will generally be much smaller than n, and that once these parameters are estimated, evaluating $\widehat{f}(\rho)$ and its derivative $\widehat{\phi}(\rho)$ is computationally very cheap.

C. Obtaining a Soft Approximation

As mentioned in Sec. II-B, greedy OD pairs that use as the link cost the derivative of its mean queue size ($\phi(\rho) = f'(\rho)$) will converge to the minimum-delay demand vector. We have so far obtained a very good approximation of the queue size (the regression function $\widehat{f}(\rho)$), and its derivative is very easy to calculate, resulting in a piece-wise constant function. However, the greedy algorithm we chose (and to the best of our knowledge, all greedy algorithms) requires a cost function with a finite relative slope (cf. item 3 in Sec. III-A), a condition that a non-continuous function such as ours does not meet. We will now discuss how to adapt our initial regression to a soft function, suitable to be used by a greedy algorithm and that has a minimum effect on the precision of the final result.

Consider that $\widehat{f}(\rho)$ is defined by n' pairs (α_i, β_i) so that $\widehat{f}(\rho) = \max_{i=1\ldots n'} \{\alpha_i + \beta_i \rho\}$. A possible approximation to this function is the so-called *log-sum-exp* function:

$$\widehat{f}^*(\rho) = \frac{1}{\gamma} \log \left(\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)} \right) \tag{7}$$

This convex non-decreasing function is clearly soft. Moreover, the precision of the approximation can be controlled by the γ parameter since:

$$\widehat{f}(\rho) \leq \widehat{f}^*(\rho) \leq \widehat{f}(\rho) + \frac{1}{\gamma} \log(n') \tag{8}$$

This means that optimizing with $\widehat{f}^*(\rho)$ as the objective function instead of $\widehat{f}$ will result in an error of at most $\log(n')/\gamma$. Actually, this bound is rather conservative, and the resulting error will generally be much smaller. As a rule of thumb, we then recommend a γ parameter that results in a maximum error of approximately 30%:

$$\gamma = \frac{\log(n')}{0.3\bar{Y}} \tag{9}$$

Where $\bar{Y}$ is the mean or median of the observations $(Y_i)_{i=1..n}$. We are now in conditions to write the final soft approximation to the link cost explicitly, which is simply the derivative of (7):

$$\widehat{\phi}^*(\rho) = \frac{1}{\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)}} \sum_{i=1}^{n'} \beta_i e^{\gamma(\alpha_i + \beta_i \rho)} \tag{10}$$

Finally, to find a value of λ that assures the convergence of REPLEX, we need to calculate an upper bound to the relative slope r of (10). Using definition 2 we see that:

$$r \geq \max_{\rho \in [0,1]} \left\{ \frac{\partial \widehat{\phi}^*(\rho)}{\partial \rho} \frac{\rho}{\widehat{\phi}^*(\rho)} \right\}$$

The derivative of $\widehat{\phi}^*(\rho)$ is:

$$\frac{\partial \widehat{\phi}^*(\rho)}{\partial \rho} = \gamma \frac{\sum_{i=1}^{n'} \beta_i^2 e^{\gamma(\alpha_i + \beta_i \rho)}}{\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)}} - \gamma \left(\frac{\sum_{i=1}^{n'} \beta_i e^{\gamma(\alpha_i + \beta_i \rho)}}{\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)}} \right)^2$$

Which, by using the first term as a bound and assuming it is a good approximation to $\gamma \phi(\rho)^2$, means that a reasonable approximation to the relative slope is:

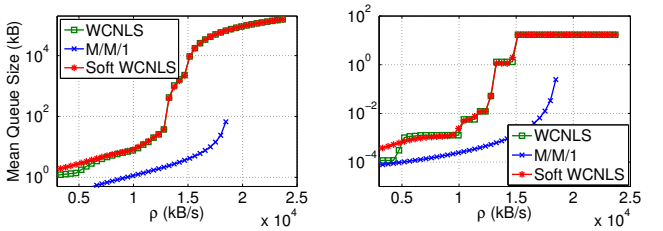
$$r \approx \max_{\rho \in [0,1]} \left\{ \gamma \phi(\rho)^2 \frac{\rho}{\widehat{\phi}^*(\rho)} \right\} \approx \gamma \max_{i=1,\ldots,n'} \beta_i \tag{11}$$

The above equation makes explicit the intuitive fact that the bigger γ is (and better the approximation) the less soft the resulting $\widehat{\phi}^*(\rho)$ is. This means that using a bigger γ than that of (9), even if it results in a smaller maximum error, translates into an insignificant improvement in precision and a decrease in the convergence speed attainable by REPLEX (cf. Sec. II-C).

IV. Simulations

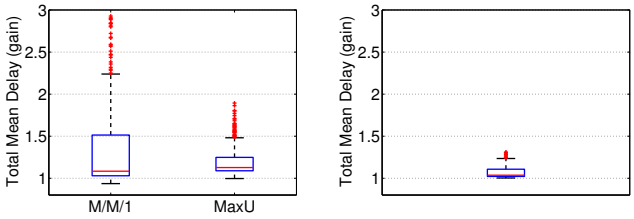
A. Assessing the Performance Gain

In the previous section we showed queue size measurements and the corresponding estimated function $\widehat{f}_l(\rho_l)$, and compared it with the *M/M/1* model, highlighting the difference between them, specially in terms of shape. If we assume this model instead of the real $f_l(\rho_l)$ we would incur in an increase of the total mean delay ($D(d)$) with respect to the optimum that may be important. To quantify this increase more precisely,



(a) $\widehat{f}(\rho)$, $\widehat{f}^*(\rho)$ and the *M/M/1* model (b) $\widehat{\phi}(\rho)$, $\widehat{\phi}^*(\rho)$ and the *M/M/1* model

Fig. 2. The regression used on the comparison



(a) M/M/1 and MaxU with respect to MinD (b) Using the $\widehat{f}_l(\rho_l)$ of the day before

Fig. 3. Increase in Total Mean Delay in the Abilene network

we will take a real network together with real demands, apply REPLEX using both the *M/M/1* model and $\widehat{\phi}^*_l(\rho_l)$ (we shall note the former as M/M/1 and the latter as MinD), and measure the difference in $D(d)$ assuming $\widehat{f}_l(\rho_l)$ as the true $f_l(\rho_l)$.

As the real topology we will take the Abilene network [16]. This academic network consists of 12 nodes and 15 bidirectional links all with the same capacity. The topology comes as an example in the TOTEM toolbox [17] and we used 395 real traffic demands (spanning a complete week) from dataset X1 from [18]. The paths we used were constructed by hand, trying to give OD pairs as much path diversity as possible, but limiting the hop count. We will assume the same $\widehat{f}_l(\rho_l)$ for all links in the network, namely the one obtained in Sec. III-B. In Fig. 2 we can see again the WCNLS regression and the resulting *M/M/1*, now together with the corresponding soft approximations (we used (9) to calculate γ). Note how, although we only used 13 of the 720 (α_i, β_i) pairs, the resulting soft approximation follows very tightly the original function. As discussed in the previous section, decreasing the number of different pairs has the double benefit of improving the precision of the soft approximation and simplifying its calculation. It is then strongly recommended to cluster the pairs (α_i, β_i) and use only the centers, for which relatively simple clustering methods may be used.

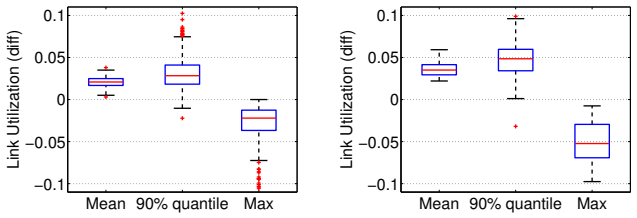
For the sake of completeness, we will also make the comparison in terms of the link utilization ($u_l = \rho_l / c_l$). A link with a u_l close to one is operating near its capacity, and in order to be able to support sudden increases in traffic and link/node failures, network operators prefer to keep links utilization relatively low. It would not make much sense to minimize the total mean delay if it meant highly utilized links. As a reference for the comparison we will use the results obtained by a greedy load-balancing mechanism whose path cost is the maximum link utilization ($\phi_P = \max_{l \in P}\{u_l\}$), which converges to a demand vector that minimizes the maximum utilization over all links [2], [3] (we shall note it as MaxU). We will measure three network-wide performance indicators: mean, 90% quantile and maximum link utilization.

In Fig. 3(a) we can see the boxplot of the results on the total mean delay. In particular, for each traffic demand, we calculated $D(d)$ for the three considered schemes (MinD, M/M/1 and MaxU), and we present the division between the value obtained by the two other schemes and ours. First of all, we can see that the total mean delay obtained by M/M/1 is generally between 5 and 50% bigger than the ones obtained by MinD. This difference may actually go as high as a 125%, and in some cases even more (although not shown for the sake of clarity of the graph, the actual maximum was 760%). If we look carefully at Fig. 2(a) we can see that this difference originates in the fact that the *M/M/1* model underestimates $f_l(\rho_l)$. In particular, the abrupt increase in queue size that occurs at $\rho \approx 13000$ kB/s, is also present in the *M/M/1* estimation, but at a much higher load of $\rho \approx 17500$ kB/s. This leads it to "believe" that links are operating at a low queueing delay load, when it is actually the opposite. On the other hand, the difference in total mean delay obtained by MaxU is generally between 10 and 25%, with a maximum of 90%. Although MaxU tries to avoid loaded links (thus obtaining better results than M/M/1), it is so conservative in its objective that it ends up unnecessarily increasing the total mean delay.

In what concerns the link utilization, we calculated the results obtained by the three considered schemes, and present the difference between the reference (MaxU) and the other two schemes, which we show in Fig. 4. It should be noted that the results for M/M/1 and MinD are very similar, except for a smaller maximum in the latter and a relatively smaller mean in the former. Quiet surprisingly, both the mean and the quantile are bigger in MaxU. The argument is the same as before. MaxU is so conservative in its objective, that, although it minimizes the maximum link utilization (where the difference with our proposal is generally less than 4%), it overlooks the less loaded links. These results confirm that minimizing $D(d)$ is a good objective, since it does not neglect links utilization. On the contrary, although it obtains a somewhat bigger maximum utilization than MaxU, all the rest of the links are more lightly loaded.

B. Temporal Behavior

A natural question that arises in our framework is how often links need to be characterized. In other words, how long can $\widehat{f}_l(\rho_l)$ be used as a good approximation of $f_l(\rho_l)$? Although more frequent updates of the links characterization will mean a more optimal or fine-tuned network, it will also mean greater computational expenses. This tradeoff between the optimality of the network and computational burden should be addressed.



(a) MaxU minus MinD (b) MaxU minus M/M/1

Fig. 4. Difference in link utilization between MaxU, MinD and M/M/1 in the Abilene network

Here we will give a partial answer to this question, and, as a reference, provide a lower bound to the validity of the link characterization used in the previous subsection (which we will note as $(\alpha_i, \beta_i)_{\text{PREV}}$). The idea is the following. From the same 72 hours long packet trace used before, we take the same 12 hours worth of measurements, but from the third day this time. We will note the characterization resulting from this new training set as $(\alpha_i, \beta_i)_{\text{NEXT}}$. We now assume that the correct $f_l(\rho_l)$ for all links in Abilene is $\widehat{f_l}(\rho_l)_{\text{NEXT}}$, and measure the increase in the total mean delay if we were to apply the load-balancing algorithm using $\widehat{\phi}_l^*(\rho_l)_{\text{PREV}}$ instead.

In Fig. 3(b) we show the boxplot corresponding to the results obtained in this case. We can see that although in some few cases the increase due to the misspecification can be more than 20%, it is generally under 10%. These results are to be compared to those obtained by the other two schemes, which obtained an excess in the total mean delay of more than 10% in half of the cases.

Our partial answer is then that the characterization of a link obtained from the measurements of any given day, is also valid the next day. It should be noted that the trace used in this study contained only working days. Our conjecture is that the characterization obtained from any working day holds for the rest of the working days in the same week. The traffic mix generally changes on weekends, which will probably result in a different $f_l(\rho_l)$ than that of the working days, thus requiring its own characterization.

C. Implementation Issues and Further Discussion

The application of our framework in a real-world network is relatively simple. Once all links have been characterized, each OD pair receives ρ_l from the links it uses (for this purpose, a TE-enabled routing protocol such as OSPF-TE may be used), calculates its paths cost with (10), and applies REPLEX to update the portion of traffic routed through each of them. This process is repeated indefinitely every some seconds. This update period should be long enough so that the quality of the obtained measurements is reasonable, but not too long to avoid unresponsiveness (in particular, we suggest 60 sec).

Regarding the learning phase (i.e. gathering the training set and performing the regression) we envisage several possibilities, differing in the degree of distribution of the resulting architecture. One possibility is that a central entity gathers the measurements, performs the regression and communicates the obtained parameters to all ingress routers (we assume that these routers, through which commodities inject traffic to the network, distribute this traffic). This first possibility presents the advantage that the required new functionalities on the router are minimal. However, as all centralized schemes, it may not be possible to implement it in some network scenarios, and handling the failure of this central entity could be very complicated. An alternative is that links (or better said, the router at the origin of the link) perform the regression. Links keep the mean queue size measurements for themselves, perform the regression and communicate the result to ingress routers. The regression could be done once a day, in the periods of low intensity (i.e. the night) so that normal operation is not affected by it. Recall that, as discussed in the previous subsection, frequent updates in the regression function are not necessary.

With respect to REPLEX, we said in Sec. II-C that the algorithm converges if the parameter λ (which should be the same for all commodities) is less than k/r (where k is a constant and r an upper bound to the relative slope of all links). The problem is how to find r, i.e. finding the biggest $\beta_{il}\gamma_l$ of all links. In the centralized architecture we described before, the problem is straightforward, since the central entity has all the information, and it should only communicate this value along with the links' characterization to all routers. In the distributed scenario it is somewhat more difficult. If routers have information of some of the links, they cannot calculate r and have to communicate with the rest to find it. Fortunately, distributed and efficient algorithms exist [19].

Concerning the training set, it is very important that the link load measurements encompass as many operating points as possible. That is to say, a good training set should include measurements from the lowest to the highest possible load. As observed in Sec. III-B, the estimated cost function is constant and underestimates the real one outside the support of the observations in the training set. This means two things. First of all, if the optimum link load is not contained in the training set, there is no guarantee that the scheme will converge to it. Finally, and even if the optimum was observed, during the convergence the algorithm may heavily load a link and "believe" the opposite if this load is outside the support of the observations for that particular link. These comments highlight the fact that measurements of a heavily loaded link (which are relatively rare) should be kept preciously and be included in all future training sets until significantly more recent measurements under similar loads become available.

A final aspect that should be highlighted is the form of the regression function $\widehat{f_l}(\rho_l)$. As we mentioned in Sec. II-A, no constraint on ρ_l was made since we assumed that $f_l(\rho_l)$ would contain such constraint implicitly, by for instance going to infinity when the link load exceeds the capacity. This is clearly not true for any $\widehat{f_l}(\rho_l)$ obtained from any training set, which means that a link may be overloaded at optimality. However, if the training set was reasonably well constructed, an overloaded link means that the network can barely support the corresponding TM independently of the

used demand vector. Anyway, and in order to enforce the constraints, we may add a non-decreasing corrective function to the approximate $\widehat{\phi}_l^*(\rho)$ that does go to infinity when the load exceeds the capacity, but is negligible at lower and more reasonable loads (for instance, we could use the *M/M/1* model for this corrective function).

V. Conclusions

In this paper we presented a Dynamic Load-Balancing (DLB) scheme that converges to the minimum total mean delay ($D(d) = \sum_l f_l(\rho_l)$, where $f_l(\rho_l)$ is the mean queue size) demand vector. The advantage of our proposal is that we make almost no a priori assumption on the function $f_l(\rho_l)$, but learn its actual form from past measurements of mean load and queue size (both readily available in most routers), thus converging to the real minimum.

The DLB algorithm was based on the known fact that the Wardrop Equilibrium of a Routing Congestion Game whose link cost is the derivative of $f_l(\rho_l)$ ($\phi_l(\rho_l) = f_l'(\rho_l)$), and if $f_l(\rho_l)$ is convex, is actually the unique minimizer of $D(d)$. We thus needed a regression method that allowed us to enforce convexity and non-decreasing constraints on the regression function, but also that its derivative could be evaluated. These requirements were fulfilled by the *Weighted Constrained Nonparametric Least Squares* problem (WCNLS), which fits a convex non-decreasing piecewise linear function to the measurements. The advantage of the weighted regression method over the original CNLS is that it can deal with heteroscedasticity and outliers, minimizing its impact on the regression. We discussed a method to assign the weights in WCNLS, which worked very well in the examples we studied. However, there are obviously many possibilities in this respect, and a thorough comparison of different methods would surely improve the framework.

In order to quantify the increase incurred by a misspecification of $f_l(\rho_l)$, we conducted a study using a real topology, several real demands and a realistic $f_l(\rho_l)$. In it, we applied the greedy algorithm with our soft approximation $\widehat{\phi}_l^*(\rho_l)$ and the derivative of the *M/M/1* model, and measured the difference in the mean total delay (assuming that our regression of $f_l(\rho_l)$ was the true delay function). The results showed that in half of the cases the increase exceeded 10%, and that it could be more than 100%. We have also studied the performance in terms of link utilization (where we took as a reference a greedy algorithm that minimized the maximum link utilization). Results indicate a very similar performance between the *M/M/1* model and our approximate $\phi_l(\rho_l)$. They both obtain a somewhat bigger maximum utilization than the optimum, but distribute the load among the rest of the links more evenly.

As discussed in Sec. I, DLB (and TE in general) is usually defined in terms of a $f_l(\rho_l)$ that is arbitrarily chosen, as long as it is convex and goes to infinity when load reaches the link capacity. In terms of link utilization and TCP performance the difference between the different $f_l(\rho_l)$ may be considered as somewhat unimportant (besides the results presented here, see [6]). However, in this paper we have shown that when the objective is minimizing queueing delays (the most important performance indicator for real-time traffic, an increasing part of traffic nowadays) this choice is crucial, and a misspecification can result in significant increases of total delay with respect to the optimum.

Another possible improvement to the framework has to do with the model used when defining $f_l(\rho_l)$. Although, as we saw in Sec. III-B, the mean queue size can be reasonably modeled with such function in wired mediums, this is not necessarily true in a wireless medium. Actually, as discussed for instance in [20], the MAC-layer interactions between routers play a significant role in determining the capacity of a link (and thus its queue size). This means that the f of any given link should include the load of all neighbor links in its collision domain, and not only itself. A deeper analysis of this non-local model also represents interesting future work.

References

[1] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *INFOCOM 2001*, vol. 3, Anchorage, USA, April 2001, pp. 1300–1309.

[2] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: responsive yet stable traffic engineering," in *ACM SIGCOMM '05*, Philadelphia, USA, 2005, pp. 253–264.

[3] S. Fischer, N. Kammenhuber, and A. Feldmann, "Replex: dynamic traffic engineering based on wardrop routing policies," in *Proceedings of the 2006 ACM CoNEXT conference (CoNEXT '06)*, Lisboa, Portugal, 2006, pp. 1–12.

[4] W. Ben-Ameur and H. Kerivin, "Routing of uncertain traffic demands," *Optimization and Engineering*, vol. 6, no. 3, pp. 283–313, september 2005.

[5] L. Kleinrock, *Queueing Systems*. Wiley-Interscience, 1975.

[6] F. Larroca and J.-L. Rougier, "Routing games for traffic engineering," in *International Conference on Communications (ICC 2009)*, Dresden, Germany, June 2009.

[7] T. Kuosmanen, "Representation theorem for convex nonparametric least squares," *Econometrics Journal*, vol. 11, no. 2, pp. 308–325, July 2008.

[8] F. Larroca and J.-L. Rougier, "Minimum-delay load-balancing through non-parametric regression," in *IFIP/TC6 Networking 2009*, Aachen, Germany, May 2009.

[9] E. Altman, T. Boulogne, R. El-Azouzi, T. Jiménez, and L. Wynter, "A survey on networking games in telecommunications," *Comput. Oper. Res.*, vol. 33, no. 2, pp. 286–311, 2006.

[10] J. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institution of Civil Engineers, Part II*, vol. 1, no. 36, pp. 352–362, 1952.

[11] S. Fischer, H. Räcke, and B. Vöcking, "Fast convergence to wardrop equilibria by adaptive sampling methods," in *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006, pp. 653–662.

[12] L. Wasserman, *All of Nonparametric Statistics: A Concise Course in Nonparametric Statistical Inference*. Springer, 2006.

[13] Y. Ait-Sahalia and J. Duarte, "Nonparametric option pricing under shape restrictions," *Journal of Econometrics*, vol. 116, no. 1-2, pp. 9–47, September-October 2003.

[14] "The MOSEK Optimization Software," http://www.mosek.com/.

[15] K. Cho, "WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18," http://mawi.wide.ad.jp/mawi/samplepoint-F/20080318/.

[16] "The Abilene Network," http://www.internet2.edu/network/.

[17] "TOTEM: TOolbox for Traffic Engineering Methods," http://totem.info.ucl.ac.be/.

[18] Yin Zhang, "Abilene Dataset," http://www.cs.utexas.edu/∼yzhang/research/AbileneTM/.

[19] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[20] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance anomaly of 802.11b," *IEEE INFOCOM 2003*, vol. 2, pp. 836–843, March-3 April 2003.