Minimum-Delay Load-Balancing through Non-parametric Regression

Federico Larroca and Jean-Louis Rougier

TELECOM ParisTech, Paris, France
46 rue Barrault F-75634 Paris Cedex 13
firstname.lastname@telecom-paristech.fr

Abstract. Network convergence and new applications running on end-hosts result in increasingly variable and unpredictable traffic patterns. By providing origin-destination pairs with several possible paths, load-balancing has proved itself an excellent tool to face this uncertainty. Formally, load-balancing is defined in terms of a convex link cost function of its load, where the objective is to minimize the total cost. Typically, the link queueing delay is used as this cost since it measures its congestion. Over-simplistic models are used to calculate it, which have been observed to result in suboptimal resource usage and total delay. In this paper we investigate the possibility of learning the delay function from measurements, thus converging to the actual minimum. A novel regression method is used to make the estimation, restricting the assumptions to the minimum (e.g. delay should increase with load). The framework is relatively simple to implement, and we discuss some possible variants.

Keywords: Traffic Engineering, Wardrop Equilibrium, Convex Non-parametric Least Squares, Next Generation Internet.

1 Introduction

As network services and Internet applications evolve, the traffic is becoming increasingly complex and dynamic. The convergence of data, telephony and television services on an all-IP network as well as user-mobility (which implies service-mobility) directly translates into a much higher variability and complexity of the traffic injected into the network. Moreover, new architectures with relatively low link capacities (such as Wireless Mesh Networks) cannot foresee overprovisioning as a viable solution. To cope with both the traffic increasing dynamism and the need for cost-effective solutions, a self-managing network architecture is required.

Dynamic load-balancing has proved itself a very efficient solution to the above issues [1, 2, 3]. 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 depends on the current demand and network condition. Since the considered time-scale is in the

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 782–794, 2009.

order of minutes, a distributed algorithm is a requirement in this kind of schemes. Mathematically, the problem is generally defined in terms of a certain convex link-cost function of the link load ($f_l(\rho_l)$), and the objective is to minimize the total network cost. For instance, if the cost function is the link utilization, the objective could be to minimize the maximum utilization on the network. Due to its simplicity, this particular objective has been considered in several routing [4] and load-balancing [2, 3] mechanisms. However, it may result in inefficient resource usage [5]. Another possible approach is to define $f_l(\rho_l)$ as a measure of the congestion in the link and minimize its sum over all links. A typical link-cost function is the queueing delay function of an *M/M/1* queue [1]. However, such simplistic model may result in bigger delays [6] and a greater maximum utilization [5] with respect to the actual minimum.

In this paper we study the possibility of designing a load-balancing mechanism that makes *no* assumptions on the delay function (except for some natural hypothesis on its shape). We will assume that $f_l(\rho_l)$ exists, but is not known, and we will estimate it from measurements. The proposed framework allows to converge to the actual minimum-delay configuration (or a very good approximation of it), thus maximizing performance. Besides, its implementation is relatively easy. The required measurements (mean incoming rate and queue size) are readily available in most routers, and the greatest upgrade required of routers is to enable load-balancing itself. Moreover, adaptations to the estimation technique are made that assure convergence of the load-balancing algorithm. The parametrization of this algorithm is also discussed.

The next section discusses the network model, our particular objective, and the distributed optimization algorithm we used. Section 3 presents the non-parametric regression algorithm we used to estimate $f_l(\rho_l)$ and some necessary adaptations to jointly use it with the distributed optimization algorithm. Implementation issues and some packet-level simulations that verify the performance of the framework are discussed in Sec. 4. The paper is concluded in Sec. 5.

2 Greedy Load-Balancing

2.1 Network Model

The network is defined as a graph $G = (V, E)$. In it there are S so-called *commodities* (or OD pairs), indexed by s and specified in terms of the triplet o_s, q_s and d_s; i.e. origin and destination nodes, and a 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}), 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 traffic distribution 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. Note that, by Little's law, $f_l(\rho_l)$ 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 is easier to measure than the queueing delay.

We are now in conditions to write the problem explicitly:

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

Note that no explicit constraint on ρ_l was made. This is assumed to be implicitly included in the link-cost 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).

2.2 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 [7]. 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, commodities are assumed to be constituted by infinitely many agents, each controlling through which path an infinitesimal amount of traffic is sent. 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 defines what is known as a *Wardrop Equilibrium* (WE) [8]. Formally, a demand vector is a WE if for each commodity $s = 1 \dots S$ 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 can be proved that a WE results in a local minimum of the so-called potential function $\Phi(d) = \sum_{l=1}^{L} \int_0^{\rho_l} \phi_l(x)dx$ [7]. This means that the WE of a congestion routing game where the link cost is the derivative of $f_l(\rho_l)$ ($\phi_l(\rho_l) = f_l'(\rho_l)$) is the solution of (1) (if $f_l(\rho_l)$ is convex, a local minimum is actually the global minimum). A distributed algorithm that converges towards such equilibrium is described in the following subsection.

2.3 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. In our case however, routers make the choice for every user (i.e. packets). It is then necessary to specify an algorithm that when independently ran in every router, the equilibrium is achieved as fast as possible and without oscillations. In [9] the authors present such mechanism and use it to design a load-balancing scheme in [3]. Below, we can see the algorithm that each commodity executes in turn (where p_{si} is the portion of demand d_s routed through path P_{si}).

1: $p'_s \leftarrow p_s$
2: **for** every pair of paths P_{si}, P_{sj} of commodity s **do**
3: **if** $\phi_{P_{si}} > \phi_{P_{sj}}$ **then**
4: $\delta \leftarrow \lambda \left((1-\beta)\, p_{sj} + \frac{\beta}{n_s} \right) \frac{\phi_{P_{si}} - \phi_{P_{sj}}}{\phi_{P_{si}} + \alpha}$
5: $p'_{si} \leftarrow p'_{si} - \delta$
6: $p'_{sj} \leftarrow p'_{sj} + \delta$
7: **end if**
8: **end for**
9: $p_s \leftarrow p'_s$

It can be seen that the portion of traffic that changes its path in a turn is proportional to the relative gain in path delay, and in a weighted mean between the portion of traffic using the new path (called *proportional sampling* in the algorithm) and $1/n_s$ (*uniform sampling*). The algorithm converges to the WE as long as $\lambda \leq k/r$, where $k > 0$ 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 [9]:

Definition 1. *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. As discussed in [3], the values of α and β are not very influential, and $\beta = 0.1$, $\alpha = 0$ turned out to be good choices.

3 Non-parametric Regression With Shape Restrictions

3.1 Convex Non-parametric Least Squares

The problem we address now is how to learn $f_l(\rho_l)$ from measurements (we are interested in its derivative, $\phi_l(\rho_l)$, but the queue size $f_l(\rho_l)$ is the observable quantity). 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 $Y_i = f(\rho_i) + \epsilon_i$ for $i = 1, \ldots, n$.

The function $f(\rho)$ is now called the *regression function* and the measurement errors $\epsilon = (\epsilon_1, \ldots, \epsilon_n)'$ are assumed to be uncorrelated random variables with $\mathbb{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2 < \infty$. The problem is to "learn" $f(\rho)$ from the observations in the training set and obtain an estimation $\widehat{f}(\rho)$. The idea is to restrict the assumptions on its functional form to the minimum. So far, we have only three necessary requirements: (i) $f(\rho)$ should clearly be increasing (ii) $\phi(\rho)$ should be non-decreasing, so $f(\rho)$ should be convex (iii) $\phi(\rho)$ should have a finite relative slope in order to make REPLEX work correctly (and probably all distributed optimization algorithms).

We will now consider the first two requirements, which are by far the most restrictive. For this, we turn our attention to the recent work of Kuosmanen [10]. Let $\mathcal{F}$ be the set of continuous, monotonic increasing and globally convex functions. The *Convex Nonparametric Least Squares* (CNLS) problem is to find $\widehat{f} \in \mathcal{F}$ that minimizes the sum of squares of the residuals:

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

Problem (2) is very difficult to solve due to the size of $\mathcal{F}$. Consider instead the following family of piecewise linear functions (where $I = \{1, \ldots, n\}$):

$$\mathcal{G}(P) = \left\{ g(\rho) = \max_{i \in I} \{\alpha_i + \beta_i \rho\} \ : \ \beta_i \geq 0;\ \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \ \ \forall j, i \in I \right\}$$

It is clear that $\mathcal{G}(P)$ belongs to $\mathcal{F}$ for any arbitrary set of observations $P = \{\rho_i\}_i$. In [10] the author proves that $\mathcal{G}(P)$ may be substituted in (2) and the same optimal solution is obtained. This result allows us to transform the infinite dimensional problem (2) into the following standard finite dimensional Quadratic Programming (QP) problem:

$$\begin{aligned} & \min_{\epsilon, \alpha, \beta} \sum_{i=1}^{n} \epsilon_i^2 \\ \text{subject to} \quad & Y_i = \alpha_i + \beta_i \rho_i + \epsilon_i \ \forall i = 1, \ldots, n \\ & \alpha_i + \beta_i \rho_i \geq \alpha_j + \beta_j \rho_i \ \forall j, i = 1, \ldots, n \\ & \beta_i \geq 0 \ \forall i = 1, \ldots, n \end{aligned} \tag{3}$$

Regarding the set of representor functions $\mathcal{G}(P)$, it may seem that a nonparametric problem was transformed 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 squared residual. Moreover, although each observation (ρ_i, Y_i) has an associated (α_i, β_i), the actual number of different values is generally a very small fraction of n. This means

that, in contrast to kernel-type regressors [11], $\widehat{f}(\rho)$ 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. Moreover, this explicit regression function allows one to intra/extrapolate with relative confidence.

Regarding (3), although it is a standard QP problem for which mature methods to solve it exist (e.g. interior point algorithms) and that several solver software are available (for instance, we used MOSEK [12]), its size is considerable. It has a total of $3n$ variables and $n(n+1)$ restrictions. The second set of constraints, which are the key to modeling convexity, are quadratic in the number of observations. The size of the problem is clearly the major drawback of the method. However, as we will discuss in Sec. 4.1, these calculations need not be performed very frequently, and they may even be delegated to a central entity.

3.2 An Example

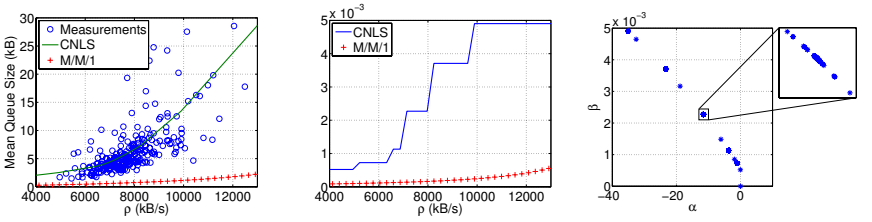
To illustrate the method, we will apply it to a training set obtained by injecting a 4 hours long packet trace (obtained from [13]) to a simple queue emulator we developed (in the absence of information on the buffer size, we assumed it infinite). The link has a capacity of 150 Mbps. Measurements are the mean queue size in kB and the mean load in kB/s over a one minute period.

Figure 1(a) shows the measurements (240 in total), $\widehat{f}(\rho)$ (as a reference, the MATLAB version of MOSEK solved (3) in less than 10 sec. in this case) and the estimation the *M/M/1* model yields $(\rho/(c-\rho))$. First of all, it should be noted that the *M/M/1* model has little to do with the real mean queue size. It consistently underestimates it, and its shape is almost a line when measurements clearly indicate a more convex curve. Regarding the estimation by CNLS, we can see that it is remarkably good, both in value and shape.

In Fig. 1(b) we show the derivative estimation through CNLS and the *M/M/1* model $(c/(c-\rho)^2)$. Since the CNLS estimation is piecewise linear, its derivative is a piecewise constant function, and after no more observations are available it becomes constant. As a consequence, CNLS will produce a good estimation of $\phi(\rho)$ in the support of the observations, after which it will systematically underestimate it. The *M/M/1* model again underestimates the derivative, except at light loads where they are both small. Finally, Fig. 1(c) shows the pairs (α_i, β_i) in the plane. As we mentioned, although there are 240 different values, they are clustered around relatively few centers. Using only these cluster centers represent an insignificant lose in precision.

3.3 How to Use CNLS for REPLEX

The final purpose of the previously described regression was to use the estimated derivative $\widehat{\phi}(\rho)$ as a cost function on REPLEX in order to obtain a good approximation to the optimal traffic distribution. Although CNLS yields a non-decreasing cost function that approximates very well $\phi(\rho)$, it presents discontinuities. This means that $\widehat{\phi}(\rho)$ has an infinite relative slope, thus making the



(a) Mean queue size measurements (b) The derivative function (c) The pairs (β_i, α_i) in the plane

Fig. 1. An example of a CNLS regression

regression method inappropriate for our purposes at first sight (cf. point (iii) in Sec. 3.1). In this subsection we discuss a possible way to approximate $\widehat{\phi}(\rho)$ through a smooth function.

Assume the regression function $\widehat{f}(\rho)$ is defined by n' (α_i, β_i) parameters so that $\widehat{f}(\rho) = \max_{i=1,\dots,n'} \alpha_i + \beta_i \rho$. A good approximation of this function is the so-called *log-sum-exp* function:

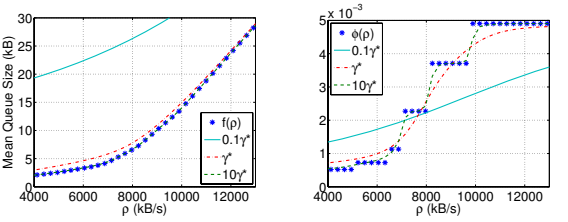
$$\widehat{f}^*(\rho) = \frac{1}{\gamma} \log \left(\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)} \right)$$

This non-decreasing and convex function is clearly smooth. Moreover, the precision of the approximation can be controlled through the parameter γ since $\widehat{f}(\rho) \leq \widehat{f}^*(\rho) \leq \widehat{f}(\rho) + \log(n')/\gamma$. This means that clustering the values of (α_i, β_i) not only decreases the size of the representation of $\widehat{f}(\rho)$, but also improves the precision of $\widehat{f}^*(\rho)$. Finally, its derivative is the following:

$$\widehat{\phi}^*(\rho) = \frac{1}{\sum_{i=1}^{n'} e^{\gamma(\alpha_i + \beta_i \rho)}} \sum_{i=1}^{n'} \beta_i e^{\gamma(\alpha_i + \beta_i \rho)} \tag{4}$$

We will use (4) as the link-cost function. A reasonable approximation to its relative slope, whose demonstration we omit for the sake of space, is $r \approx \gamma \max_{i=1,\dots,n'} \beta_i$. This formula makes explicit the intuitive fact that the bigger γ is (and better the approximation) the less soft the resulting $\widehat{\phi}^*(\rho)$ is.

The problem now is how to assign γ. As a rule of thumb, we recommend using a value such that the error in the soft approximation is approximately 30%. That is to say, $\gamma = log(n')/(0.3\bar{Y})$ where $\bar{Y}$ is the mean of Y. This value, as we shall now illustrate with an example, results in a good tradeoff between precision and convergence speed. We will consider the same $\widehat{f}(\rho)$ as in Fig. 1, and use three different values of γ: our recommended 30% error value (γ^*), $10\gamma^*$ and $0.1\gamma^*$. In Fig. 2 we can see the resulting $\widehat{f}^*(\rho)$ and $\widehat{\phi}^*(\rho)$ for the three considered values. First of all, a value as small as $0.1\gamma^*$ results in too much error for all



(a) $\widehat{f}(\rho)$ and $\widehat{f}^*(\rho)$ for some values of γ (b) $\widehat{\phi}(\rho)$ and $\widehat{\phi}^*(\rho)$ for some values of γ

Fig. 2. An example of approximating a piecewise linear function with the log-sum-exp

practical purposes. On the other hand, the difference between γ^* and $10\gamma^*$ is almost insignificant for $\widehat{f}^*(\rho)$, and reasonably small for $\widehat{\phi}^*(\rho)$. Moreover, note that γ^* results in a ten times smaller relative slope than $10\gamma^*$, thus allowing a convergence speed ten times faster (cf. 2.3).

4 Simulations

4.1 Implementation Discussion

The application of our framework in a real-world network is relatively simple. Once all links have been characterized (i.e. we have the parameters $(\alpha_i, \beta_i)_l$ for all l), each OD pair receives ρ_l from the links it uses[1], calculates its paths cost with (4), and applies REPLEX to update its traffic distribution. This process is repeated indefinitely every some seconds (in particular, we used 60 sec). This update period should be long enough so that the obtained measurements' quality is reasonable, but not too long to avoid unresponsiveness.

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 and on what data is used at each moment.

With respect to who does which calculations, 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 has 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 ingress routers perform the regression. Links measure their load and mean queue size, communicate periodically these measurements to all ingress routers (instead of the central entity), which in turn perform the regression. However, the regression for any given link would be performed by several routers, constituting a waste of resources. A more reasonable alternative is that links (or better said, the router at the origin of the link)

[1] For this purpose, a TE-enabled routing protocol such as OSPF-TE may be used.

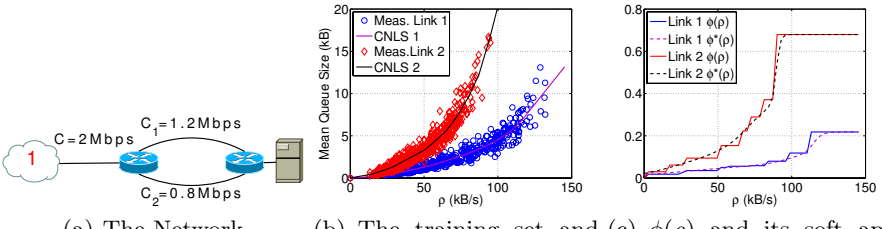
perform the regression. Links keep the mean queue size measurements for themselves, perform the regression and communicate the result to ingress routers. The regression could be done once a day, in the periods of low intensity (i.e. the night) so that normal operation is not affected by it.

A second aspect that has different possibilities is what characterization (i.e. $(\alpha_i, \beta_i)_l$) use at each moment. For instance, measurements could be gathered every day, the regression performed, and its result used the next day. Another possibility is to use the result of the measurements of the same day the previous week. More granularity could be added, and we could use different characterizations for different moments of the day. What granularity is needed and if it is actually necessary is an analysis that we let for future work.

4.2 Examples

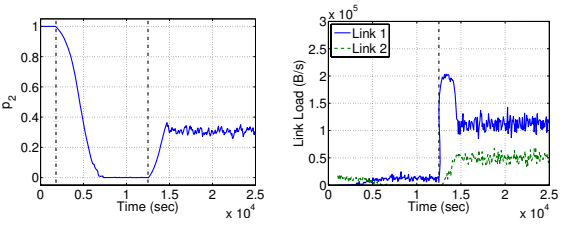
In this subsection we will consider two relatively simple examples we implemented in ns-2 [14] that will help us gain further insight into the framework and verify its correctness in the presence of delayed and noisy measurements. We will assume that the training set has been obtained and the regression performed by a central entity. Routers only have the pairs $(\alpha_i, \beta_i)_l$, their associated γ_l (which was calculated with the formula discussed in Sec. 3.3) and already know λ (cf. Sec. 2.3). As a final remark, in all the simulations load balancing is performed at the granularity of flows (i.e. once a flow is routed through a path, it rests there throughout its lifetime) and is random (i.e. new incoming flows are routed through path P_{si} with probability p_{si}).

We will begin with the simplest example: one commodity has two one-hop paths (see Fig. 3(a)). Traffic is a mixture of elastic and streaming flows. The elastic ones (whose size is exponentially distributed with mean 20 kB) are generated as a Poisson process. The streaming part of traffic is constituted of CBR flows (at a bitrate of 10 kbps and an exponentially distributed duration with mean 20 sec.) also arriving as a Poisson process, and it represents 10% of the total traffic. The measurements (467 for each link) were obtained by averaging the link load and the queue size over a minute period. In Fig. 3(b) we can see the measurements together with the resulting regression, and in Fig. 3(c) we show the corresponding cost function $\phi_l(\rho_l)$ and its soft approximation $\phi_l^*(\rho_l)$.



(a) The Network (b) The training set and the regression (c) $\phi(\rho)$ and its soft approximation $\phi^*(\rho)$

Fig. 3. The single-source case example topology and regression

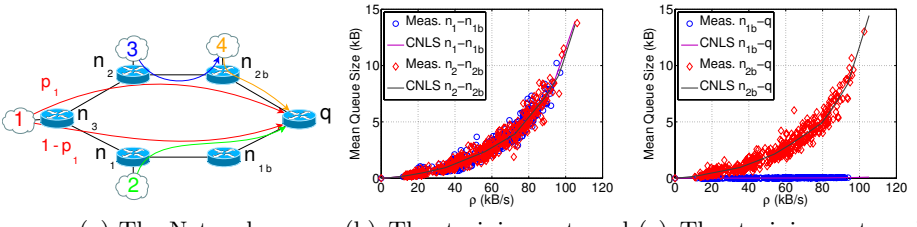


(a) p_2 as a function of time (b) The links load

Fig. 4. Simulation results in the single-source case example

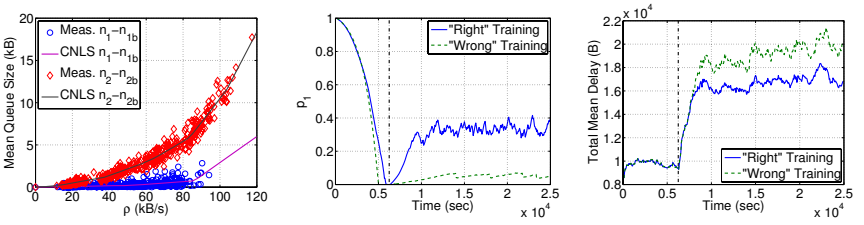
Using this topology we will run a two-parts simulation. In its first half the total demand is approximately 100 kbps, after which it abruptly increases to 1200 kbps (this moment is marked in Fig. 4 by a vertical line). The traffic distribution is updated after second 1800 (marked by the first vertical line in Fig. 4(a)). Notice in Fig. 4(a) how at first p_2 (the portion of traffic routed through the lower path) changes relatively slow, but as it decreases the change grows faster. This is a consequence of the sampling step of REPLEX, in particular of the proportional one. The small (but inevitable) oscillations around the optimum traffic distribution should also be noted. They are inevitable since ρ_l measurements are noisy (see Fig. 4(b)), but the effect of this noise on the convergence is minimized by the algorithm. Finally, notice how at the beginning of the second half of the simulation load on link 1 momentarily goes to values outside the support of the training set. As discussed in Sec. 3.2, for all such values of ρ, $\phi(\rho)$ is constant when it should actually increase. Although this does not prevent the algorithm from reaching the optimum, the convergence speed is slower than it should, resulting in an overloaded link for almost 30 iterations (or 30 minutes in our case). Although such an abrupt increase in traffic should be rare to say the least, this highlights the importance of a training set that encompasses as much operational points as possible.

We will now consider a somewhat more complex case scenario. The network (see Fig. 5(a)) consists of six links, all with a capacity of 1 Mbps. There are a total of 4 commodities whose destination is the same node q, but only commodity



(a) The Network (b) The training set and the regression for link n_1-n_{1b} and n_2-n_{2b} (c) The training set and the regression for link n_{1b}-q and n_{2b}-q

Fig. 5. The second example: two paths and four commodities



(a) Alternative training set and regression for link n_1-n_{1b} and n_2-n_{2b}

(b) p_1 as a function of time

(c) The total mean delay $D(d)$ as a function of time

Fig. 6. An alternative training set for the second example

1 can use more than one path. The traffic generated by each commodity has the same characteristics as in the previous example.

In Fig. 5(b) and Fig. 5(c) we can see the training set and the corresponding regression for four of the links. The training set was obtained by changing the traffic intensity of the four commodities at the same time (with p_1 fixed at 0.5). Notice how the mean queue size of link n_{1b}-q is near zero regardless of its load, and how its "symmetric" link (n_{2b}-q) is quiet the opposite and may be considered identical to n_2-n_{2b}. Traffic through link n_{1b}-q does not generate significant queue because link n_1-n_{1b} already shaped the traffic.

This example introduces the problem of links that have an insignificant queue size independently of its load. This may be due to traffic characteristics (as before), or because the link buffer is small. A link of such characteristics clearly presents a problem for our framework. For instance, let us consider an alternative training set, obtained in a situation where commodity 2 sends little or no traffic and commodity 1 sends most of its traffic through the lower path. In Fig. 6(a) we can see that although the small amount of traffic generated by commodity 2 results in a little bit of queue in link n_1-n_{1b}, the mean queue size $f_l(\rho_l)$ (and thus $\phi_l(\rho_l)$) for this link is almost zero except at big loads.

Consider now the following situation. Commodities 2, 3 and 4 all generate the same demand (approximately 450 kbps). Commodity 1 generates approximately 100 kbps during the first fourth of the simulation, and then abruptly increases its demand to the same value as the rest (this moment is marked with a vertical line in Fig. 6). Figure 6(b) shows the evolution of p_1 over time when using the results of both training sets. In the first part they both converge to $p_1 = 0$, but when d_1 increases we can appreciate the difference between the training sets. While the "right" training set (i.e. the one of Fig. 5(b)) moves towards a reasonable $p_1 = 0.4$, the other training set gets stuck in $p_1 = 0.05$ which results in a higher total delay (see Fig. 6(c)) and two almost overloaded links (n_1-n_{1b} and n_{1b}-q).

5 Concluding Remarks

In this paper we presented a dynamic load-balancing mechanism that converges to an excellent approximation of the minimum-delay traffic distribution without

assuming any given delay model. This was achieved on the one hand by "learning" the delay function from measurements, and on the other hand by applying a greedy load-balancing algorithm with provable convergence (and verified by our packet-level simulations). The chosen regression method is a piecewise linear fitting method, where the number and position of the lines are endogenously determined to minimize the squared residual. The cost function, which is the derivative of this regressor, was then not continuous, a fact that poses a problem to the distributed algorithm. This forced us to make a soft approximation of the regressor function, controlled by a single parameter γ, for which we gave hints on how to assign it. The few parameters the distributed algorithm requires were also discussed in the paper.

In Sec. 4 we highlighted two shortcomings of our framework which may result in overloaded links: the regression outside the support of the observations is not reliable (since the cost function does not increase any further) and links that present little or no queueing delay always have a negligible cost. A possible solution to both problems is adding to the link cost a known parametric function that is negligible with respect to $\phi_l(\rho_l)$ except at very high loads. This will increase the cost of loaded links both when no observations are available or when their actual mean queueing delay is small. Moreover, if the optimum does not load considerably any link, it will be attained with a very small error. Further development of such correction is the subject of future work.

It would also be very interesting to perform a deeper statistical analysis of the behavior of the mean queue size with respect to load. A possible analysis would be to study how often does the regression function change over time (i.e. answer the question of whether the mean queue size function changes over time, and how often it does).

Regarding the queueing model, we considered that the mean queue size is a function of the mean incoming rate only. This is naturally not true, as it actually depends on the complete packet arrival process. Methods that estimate ϕ_l considering the whole process exist, such as the one used in [6]. Apart from being more complicated (they require to measure the arrival and departure time of every packet), the problem with such methods is that ϕ_l now depends on a number of unknown and uncontrollable variables. This results in the impossibility of guaranteeing convergence to the optimum by changing the portions of traffic only, and it does result in oscillations as presented (but not explained) in [6]. A possible improvement to our model is to consider that ϕ_l depends on the mean load of its incoming links. For instance, in Fig. 5(a), that the mean queueing delay in link n_1-n_{1b} depends on the load on links n_3-n_1 and the one connecting commodity 2 and n_1. For this now multi-dimensional regression problem the same method may be used. A deeper analysis of this alternative model represents also interesting future work.

References

1. Elwalid, A., Jin, C., Low, S., Widjaja, I.: MATE: MPLS adaptive traffic engineering. In: INFOCOM 2001, vol. 3, pp. 1300–1309 (2001)
2. Kandula, S., Katabi, D., Davie, B., Charny, A.: Walking the tightrope: responsive yet stable traffic engineering. In: ACM SIGCOMM 2005, pp. 253–264 (2005)
3. Fischer, S., Kammenhuber, N., Feldmann, A.: Replex: dynamic traffic engineering based on wardrop routing policies. In: CoNEXT 2006, pp. 1–12 (2006)
4. Ben-Ameur, W., Kerivin, H.: Routing of uncertain traffic demands. Optimization and Engineering 6(3), 283–313 (2005)
5. Larroca, F., Rougier, J.L.: A fair and dynamic Load-Balancing mechanism. In: International Workshop on Traffic Management and Traffic Engineering for the Future Internet (FITRAMEN) 2008, Porto, Portugal (December 2008)
6. Cassandras, C., Abidi, M., Towsley, D.: Distributed routing with on-line marginal delay estimation. IEEE Trans. Comm. 38(3), 348–359 (1990)
7. Altman, E., Boulogne, T., El-Azouzi, R., Jiménez, T., Wynter, L.: A survey on networking games in telecommunications. Comput. Oper. Res. 33(2), 286–311 (2006)
8. Wardrop, J.: Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil Engineers, Part II 1(36), 352–362 (1952)
9. Fischer, S., Räcke, H., Vöcking, B.: Fast convergence to wardrop equilibria by adaptive sampling methods. In: STOC 2006: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, pp. 653–662 (2006)
10. Kuosmanen, T.: Representation theorem for convex nonparametric least squares. Econometrics Journal 11(2), 308–325 (2008)
11. Wasserman, L.: All of Nonparametric Statistics: A Concise Course in Nonparametric Statistical Inference. Springer, Heidelberg (2006)
12. The MOSEK Optimization Software, `http://www.mosek.com/`
13. Cho, K.: WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18, `http://mawi.wide.ad.jp/mawi/samplepoint-F/20080318/`
14. The Network Simulator - ns, `http://nsnam.isi.edu/nsnam/index.php/Main_Page`