Computer Networks 56 (2012) 1152–1166

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Minimum delay load-balancing via nonparametric regression and no-regret algorithms

Federico Larroca *, Jean-Louis Rougier

Telecom ParisTech, 46 rue Barrault, F-75634 Paris Cedex 13, Paris, France

ARTICLE INFO

Article history:
Received 23 March 2011
Received in revised form 25 November 2011
Accepted 30 November 2011
Available online 8 December 2011

Keywords:
Wardrop equilibrium
Convex nonparametric least squares
Weighted least squares
No-regret

ABSTRACT

In the current network scenario, where traffic is increasingly dynamic and resource demanding, Dynamic Load-Balancing (DLB) has been shown to be an excellent Traffic Engineering tool. In particular, we are interested in the problem of minimum delay load-balancing. That is to say, we assume that the queueing delay of a link is given by a function of its load. The objective is then to adjust the traffic distribution over paths so that, for the current traffic demand, the addition of these functions times the load is minimized. The contribution of our article is twofold. Firstly, we analyze the possibility of using so-called *no-regret* algorithms to perform the load balancing. As opposed to other distributed optimization algorithms (such as the classical gradient descent) the algorithm we discuss requires no fine-tuning of any speed-controlling parameter. Secondly, we present a framework that does not assume any particular model for the queueing delay function, and instead learns it from measurements. This way, the resulting mean delay of optimizing with this learnt function is an excellent approximation of the real minimum delay traffic distribution. The whole framework is illustrated by several packet and flow level simulations.

1. Introduction

Two aspects of the current networking scenario call for new and more effective ways of handling traffic. First of all, service convergence in the same network, in addition to the ever increasing access rates available for end-users, have led to very dynamic and unpredictable traffic patterns. Secondly, the assumption that overprovisioning is the panacea for all problems is being increasingly reconsidered. This is due on the one hand to the emergence of new architectures with intrinsically scarce resources (e.g. Wireless Mesh Networks), and on the other hand to the fact that the assumption that core capacities are orders of magnitude higher than access rates may no longer hold in the near future.

Robust Routing (RR) [1–3] has emerged in recent years as a possible answer to the above problems. In RR, traffic uncertainty is taken into account directly within the routing optimization, computing a single routing configuration for all traffic demands within an *uncertainty set* where traffic is assumed to vary. This uncertainty set can be defined in different ways, depending on the available information; e.g. largest values of link loads previously seen, a set of previously observed demands (previous day, same day of the previous week, etc.). Although relatively simple and naturally stable, RR presents some important drawbacks. Firstly, the definition of the set of traffic demands presents a clear tradeoff: larger sets are able to handle a broader group of traffic demands, but at the cost of routing inefficiency; conversely, tighter sets produce more efficient routing schemes, but are subject to poor performance guarantees [3]. Secondly, optimization under uncertainty

* Corresponding author. Address: Facultad de Ingeniería, Universidad de la República, Julio Herrera y Reissig 565, C.P., 11300 Montevideo, Uruguay. Tel.: +598 2 711 09 74; fax: +598 2 711 74 35.
E-mail addresses: flarroca@fing.edu.uy (F. Larroca), rougier@telecom-paristech.fr (J.-L. Rougier).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.11.015

is generally more complex than classical optimization, which forces the use of simpler optimization criteria. In this sense, the typical objective is to minimize the maximum link utilization (MLU) in the network. Other more complex objective functions (such as the one considered in this article) generally result in a marginally bigger MLU and an improvement in other performance indicators (such as the mean link utilization) that may be significant (see for instance [4,5] or the results we present here).

To deal with these drawbacks, *Dynamic Load-Balancing* (DLB) has been proposed [5–9]. In these schemes each Origin–Destination (OD) pair is connected by several, established a priori, paths. Based on feedback from the network, each OD pair dynamically adjusts the portion of its traffic sent through each path. The objective is, given the configured paths and the current traffic demand, to minimize a certain network-wide cost function. Although this "always optimized" characteristic is very attractive, the deployment of DLB has been, to say the least, limited. Network operators are reluctant to use dynamic mechanisms mainly because they are afraid of a possible oscillatory behavior of the algorithm used by each OD pair to adjust their traffic distribution (as the early experiences in ArpaNet has proved [10], these concerns are not without reason). Indeed, previously proposed DLB algorithms always include a parameter that controls the convergence speed, which is very tricky to assign. Although for each algorithm there exists a range for this parameter in which it is stable, these values result in unresponsiveness in certain situations.

Our first contribution is to present a Dynamic Load-Balancing algorithm based on so-called *no-regret* algorithms. The authors of [11] proved that OD pairs using algorithms of this kind converge to a greedy equilibrium which may be induced to coincide with the optimum we are looking for (see the following sections for a more detailed discussion). In particular, we will use a variation of the *Incrementally Adaptive Weighted Majority algorithm* [12], which presents the advantage of being completely self-regulated, thus avoiding any tricky speed-controlling parameter setting and the reactivity–stability tradeoff we mentioned before. Furthermore, we will present certain adaptations of the algorithm that are necessary to enable its use in the presence of non-stationary traffic.

In most DLB schemes the objective function is the addition over all links of a certain link-cost function. The idea is that this function should measure 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 types of traffic) and simple algebra (the total delay of a path is the addition of the delay at each link). However, most DLB schemes require an analytical expression of this delay, for which classic and oversimplistic models (e.g. $M/M/1$) are used [6], resulting in an actual total delay that is significantly bigger than the optimum.

As a second contribution of this article, we propose a framework that makes very few assumptions on the delay function. Except for some natural hypothesis on its shape (e.g. monotonicity) we will only assume that the queueing delay on link l is of the form $f_l(\rho_l)$ (i.e. depends only on the mean load of the link ρ_l). The actual form of $f_l(\rho_l)$ will be obtained (or learned) from past measurements. To achieve this we present two different regression methods. The first one is a variation of the nonparametric regression method presented in [15], and finds the regression function that best fits the measurements. However, it presents scalability issues as the number of available measurements increases. We consider then the algorithm presented in [16]. This heuristic finds a parametric function that reasonably adjusts the measurements in a very short time, although its precision is not as good as the one obtained by the first method.

The complete framework is illustrated by several flow as well as packet level simulations using a real topology and several real traffic demands. For instance, our study indicates that using the $M/M/1$ model instead of our learned function results in an increase in the total queueing delay that may easily exceed 10%, and can go as high as more than 80%. 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 article is structured as follows. In the next section we define the network model and discuss the resulting equilibrium of greedy users. The algorithm used to converge to this equilibrium, when the delay function is known, will be presented in Section 3. In Section 4 we present the two methods to obtain a robust approximation of the delay function from past measurements. We make a performance analysis of our framework in Section 5, where we also present the final form of the load-balancing algorithm. In Section 6 we present some packet-level simulations and discuss the implementation issues that arise from our framework. Related work is discussed in Section 7. We conclude the article in Section 8.

2. Greedy load-balancing

2.1. 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 $s = 1,\ldots,S$, 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. Commodity s can use any path from set $\mathcal{P}_s$, where each of its elements (noted as P_{si} with $i = 1,\ldots,n_s$) is a subset of E connecting o_s to q_s. In practice, paths are chosen to be no less than two per commodity, and to differ as much as possible for resiliency reasons. In particular, we used the method presented in [26] to choose these paths.

All commodities can distribute their total demand arbitrarily along their paths. In particular, commodity s sends an amount $d_{P_{si}}$ of its traffic through path P_{si}, where $d_{P_{si}} \geqslant 0$ and $\sum d_{P_{si}} = d_s$. This distribution of traffic induces the demand vector $d = (d_{P_{si}})$.

Given the demand vector, the total load on link l is then $\rho_l = \sum_s \sum_{P\in\mathcal{P}_s : l\in P} d_P$. 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 the measure of the network total congestion we shall use the *mean end-to-end queueing*