12th GI/ITG CONFERENCE ON MEASURING, MODELLING AND EVALUATION OF COMPUTER AND COMMUNICATION SYSTEMS
3rd POLISH-GERMAN TELETRAFFIC SYMPOSIUM

ADAPTIVE LOAD BALANCING USING MPLS

Riikka Susitaival, Samuli Aalto, and Jorma Virtamo
Networking Laboratory, Helsinki University of Technology
P.O.Box 3000, FIN-02015 HUT, Finland
Email: {riikka.susitaival, samuli.aalto, jorma.virtamo}@hut.fi

Abstract

MPLS brings up new possibilities to improve the performance of IP networks. Notably the explicit routing of MPLS facilitates balancing the load by moving traffic from a congested part of the network to some other part in a well controlled way. In this paper we study adaptive load balancing based on measured link loads without knowledge of the full traffic matrix. The objective for load balancing can be, e.g. minimizing the maximum link utilization or minimizing the mean delay in the network. We present a simple adaptive and distributed algorithm to obtain these objectives. In addition, a numerical method is developed to evaluate the performance of the algorithm. The method is applied to three test networks. The results show that the maximum link utilization and the mean delay obtained in a reasonable number of iterations are very close to the optimal values.

Keywords
MPLS, Traffic Engineering, adaptive routing, load balancing

1 INTRODUCTION

In the current Internet each router forwards packets independently based on packet's destination address and precalculated routing tables. Multi Protocol Label Switching (MPLS) makes route control more effective and enables new services [1]. A significant applicationof MPLS is Traffic Engineering (TE), which concerns performance optimization of operational networks. MPLS provides capabilities to predefine the paths used between each ingress and egress node. It also enables splitting the traffic into several paths instead of one single path used by traditional routing protocols. Explicit routing and traffic splitting give the ability to balance load and thus improve the performance of the network.

Load balancing can be based on long term traffic averages or shorter term measured traffic conditions. The objective can be minimizing the mean delay or minimizing the maximum link utilization, for example. The first one emphasizes both load balancing and short paths, whereas the second one can route traffic along long routes also. In IP networks, minimization of mean flow response times is actually more important than minimization of mean packet delays. However, as packet delays are easier to analyze we focus on them. One can expect that, as optimization criteria, they are similar.

A static version of the load balancing problem, in which the traffic matrix describing the long term traffic averages is assumed to be known, has been studied for a long time, see e.g. [2]. However, if such information is not available, or traffic conditions change unexpectedly, another approach is needed. One possibility is to use adaptive methods in which load is balanced based on measurements, such as end-to-end monitoring or monitoring of each link individually. Examples of such methods are given below.

Elwalid et al. [3] introduce a mechanism called MPLS Adaptive Traffic Engineering (MATE). MATE is a state-dependent traffic engineering mechanism, in which the traffic load is balanced using a distributed adaptive algorithm. The algorithm tries to equalize congestion measures among the LSPs by approximating the gradient-projection algorithm, which transfers traffic toward the direction of the gradient projection of the objective function. Song et al. [4] introduce a concept called Load Distribution over Multipath (LDM), in which traffic is split dynamically at flow level into multiple paths. The LSP for the incoming traffic is selected from a fixed set randomly based on congestion and the length of the path.

We find some drawbacks in the algorithms introduced above. First, in [3] traffic engineering is based on the measured traffic conditions between each ingress and egress pair. However, these measures offer overlapping information since LSPs might use same links. The amount of control data can be reduced by using only link load information. Second, momentary efficiency of the algorithm in [4] depends largely on the flow-level dynamics. It remains unclear whether the granularity of the traffic splitting at the flow level is fine enough to provide stable network conditions.

In this paper we suggest a simple adaptive and distributed load balancing algorithm that continuously makes incremental changes in the load distribution based on measured link loads. The changes are made by the edge routers independently of each other. In addition, a numerical method is developed to evaluate the performance of the load balancing algorithm. The method is applied to three test networks.

The paper is organized as follows. In Section 2, we review the static load balancing problem, which is transformed to a corresponding dynamic problem in the absence of information on the long-term traffic demands in Section 3. In Section 4, we describe our adaptive and distributed algorithm for the dynamic load balancing problem. The performance of the algorithm is numerically evaluated in Section 5. Section 6 concludes the paper.

2 STATIC LOAD BALANCING PROBLEM

In this section we review the well-known static load balancing problem[1] [2, 3] applied to the MPLS context.

[1] Also known as optimal routing problem.

Consider a network consisting of nodes $n \in \mathcal{N}$ and links $l \in \mathcal{L}$. Let c_l denote the capacity of link l. The network is loaded by traffic demands d_k between ingress-egress (IE) node pairs $k \in \mathcal{K}$. These demands should be understood as long-term time-averages, and not as instantaneous values. Let s_k denote the ingress node and t_k the egress node of IE-pair k. Ingress and egress nodes are edge routers of the MPLS network under consideration. Each IE-pair k has a predefined set $\mathcal{P}_k$ of LSP's. Let $\mathcal{P} = \cup_{k\in\mathcal{K}}\mathcal{P}_k$ denote the set of all possible paths. Each path $p \in \mathcal{P}_k$ consists of a concatenation of consecutive links from s_k to t_k. We use notation $l \in p$ whenever link l belongs to path p.

Let x_p denote the rate of traffic allocated to path p. The x_p are the control variables in our static load balancing problem. Given the x_p, the induced load y_l on link l is

$$y_l = \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k : l\in p} x_p, \quad \text{for all } l \in \mathcal{L}.$$

It is easy to see that, in general, the mapping between the traffic demands d_k and the link loads y_l is not one-to-one, i.e. while d_k's determine y_l's uniquely, the opposite is not true. Load y_l on link l incurs cost $C_l(y_l)$, which is assumed to be an increasing and convex function of the load y_l.

The objective in the *static load balancing problem* is to minimize the total cost by choosing an optimal traffic allocation $x^* = (x_p^*; p \in \mathcal{P})$. The problem is formulated as follows:

$$\begin{array}{lll} \text{Minimize} & C(x) = \sum_l C_l(y_l) & \\ \text{subject to} & y_l = \sum_{k\in\mathcal{K}} \sum_{p\in P_k : l\in p} x_p, & \text{for all } l \in \mathcal{L}, \\ & \sum_{p\in P_k} x_p = d_k, & \text{for all } k \in \mathcal{K}, \\ & x_p \geq 0, & \text{for all } p \in \mathcal{P}. \end{array} \tag{1}$$

A typical cost function is as follows:

$$C_l(y_l) = \frac{y_l}{c_l - y_l}. \tag{2}$$

By this choice the total mean delay is minimized, assuming that the MPLS network can be modelled as a Jackson queueing network.

The partial derivative of the total cost function $C(x)$ with respect to x_p satisfies

$$\frac{\partial C(x)}{\partial x_p} = \sum_{l\in p} C_l'(y_l).$$

It is called the *first derivative length* of path p. An optimal allocation of traffic demand d_k is such that $x_p^* > 0$ only for those paths $p \in P_k$ that minimize the first derivative length. A standard technique to solve the static load balancing problem is the gradient projection algorithm.

In another formulation of the static load balancing problem the maximum link cost is minimized (instead of the total cost):

$$C(x) = \max_{l\in\mathcal{L}} C_l(y_l). \tag{3}$$

In this case a typical cost function is

$$C_l(y_l) = \frac{y_l}{c_l}. \tag{4}$$

By this choice the maximum link utilization is minimized. We note that the minimization of the maximum link utilization may result in long paths consuming much resources, while the minimization of the total mean delay both balances the load and gives preference to short paths.

The static problem is possible to be formulated and solved only if we have precise information on all the traffic demands d_k. It may well be the case that such information is either imprecise, outdated, or totally missing. In such a case, another approach is needed. In the following sections we first formulate such a dynamic load balancing problem and then describe an adaptive and distributed algorithm that does not need any information on traffic demands d_k to solve the dynamic problem.

3 DYNAMIC LOAD BALANCING PROBLEM

In this section we consider the dynamic load balancing problem with incomplete information. More precisely, we assume that the traffic demands are unknown but the link loads are periodically measured using a measurement system as in [5]. As noted in the previous section, one cannot derive the traffic demands from the link loads [2]. Therefore, the problem cannot be formulated, let alone solved, as a static load balancing problem.

As before, consider a network consisting of nodes $n \in \mathcal{N}$ and links $l \in \mathcal{L}$. The network is loaded by the traffic demands between IE-pairs $k \in \mathcal{K}$, but these demands are unknown. Instead, we assume that the link loads are measured periodically at times t_i. These measured link loads $\hat{y}_l(i)$ should be understood as time-averages over the whole measurement period (t_{i-1}, t_i), and not as instantaneous values. On the other hand, due to traffic fluctuations in different time scales, the measured loads deviate from the long-term time averages in a random manner (we will later assess the effect of the traffic fluctuations by numerical experiments). We assume that the information on the measured loads is distributed to all edge routers in the MPLS network. This can be done in a similar way as the link states are distributed to all routers within an AS in OSPF. We further assume that the time needed to distribute the information to edge routers is negligible in comparison to the length of the measurement period.

Each IE-pair k has a predefined set $\mathcal{P}_k$ of LSP's. Let $\phi_p(i)$ denote the *proportion* of traffic allocated to path p at time t_i. These splitting ratios are the control variables in our dynamic load balancing problem satisfying, for all k and i,

$$\sum_{p \in \mathcal{P}_k} \phi_p(i) = 1.$$

[2] To be precise, the problem of traffic matrix estimation is strongly underdetermined and requires prior assumptions to get a solution

Splitting ratios $\phi_p(i)$ are gradually adjusted based on the measured link loads $\hat{y}_l(i)$. Thus, the algorithms to determine these ratios may be called *adaptive*.

It is important to note that, since these splitting ratios determine the proportion of traffic, and not the absolute amount of traffic, allocated to LSP's, there is no need to know the traffic demands. This kind of traffic splitting may be done at the packet level or at the flow level. If traffic splitting is done at the packet level, reordering of packets may be required. Splitting traffic at the flow level avoids this problem, but then poor granularity may be a drawback.

Assuming that the traffic demands are constant (albeit unknown), the objective in the *dynamic load balancing problem* is to choose, after each measurement period (t_{i-1}, t_i), the splitting ratios $\phi(i) = (\phi_p(i); p \in \mathcal{P})$ in such a way that they converge, as soon as possible, to the unknown optimal values ϕ_p^* of the corresponding static load balancing problem.

4 AN ADAPTIVE AND DISTRIBUTED LOAD BALANCING PROBLEM

In this section we describe an adaptive and distributed algorithm to solve the dynamic load balancing problem presented in the previous section. The algorithm is heuristic, and thus suboptimal. However, it seems to work quite well, as we will see in the following section, where the performance of the proposed algorithm is evaluated numerically.

First we note that, since the measured loads $\hat{y}_l(i)$ are distributed to all edge routers, the decisions concerning the splitting ratios $\phi_p(i)$ can be done in a distributed way. Thus, in our *distributed* algorithm, each edge router independently determines the splitting ratios $\phi_p(i)$ for all those paths p for which it is the ingress node.

Next we describe how an edge router determines the splitting ratios for any of the IE-pairs for which it is the ingress node. Consider one such IE-pair, say k. As our algorithm is *adaptive*, the splitting ratios will depend on the measured loads $\hat{y}_l(i)$. More precisely, they will depend on the estimated path costs $D_p(\hat{y}(i))$, where $\hat{y}(i) = (\hat{y}_l(i); l \in \mathcal{L})$ denotes the vector of measured loads. The definition of the path cost $D_p(\hat{y}(i))$ depends on the formulation of the original optimization problem. If the purpose is to minimize the total cost (1), such as the mean delay, then we define two different path costs. One is the total cost along the path,

$$D_p(\hat{y}(i)) = \sum_{l \in p} C_l(\hat{y}_l(i)), \tag{5}$$

and the other one is its first derivative length,

$$D_p(\hat{y}(i)) = \sum_{l \in p} C_l'(\hat{y}_l(i)), \tag{6}$$

But if the objective is to minimize the maximum cost (3), such as the maximum utilization, then the path costs are naturally defined by

$$D_p(\hat{y}(i)) = \max_{l \in p} C_l(\hat{y}_l(i)) \tag{7}$$

The idea is simply to alleviate the congestion on the most costly path (among the paths belonging to the same $\mathcal{P}_k$) by reducing its splitting ratio. This should, of course, be compensated by increasing the splitting ratio of some other path within the same set $\mathcal{P}_k$. We study two different rules: (a) choose the other path randomly among the other paths, or (b) choose a path with minimum path cost.

Since the algorithm is adaptive, we have a closed-loop control problem: the splitting ratios that depend on measured loads have a major effect on the upcoming load measurements. It is well known that feedback control systems are prone to instability if the gain in the loop is too large. Thus, to avoid harmful oscillations, we let the splitting ratios change only with minor steps. The step size is determined by the granularity parameter g that will be defined below. A finer granularity is achieved by increasing the value of g. We will study an appropriate choice of g by numerical experiments in Section 5.

Algorithm to determine the splitting ratios. At time t_i, after receiving the information concerning all the measured loads $\hat{y}_l(i)$, the edge router s_k operates as follows:

1. Calculate the path costs $D_p(\hat{y}(i))$ for each path $p \in \mathcal{P}_k$.
2. Find the path $q \in \mathcal{P}_k$ with maximum cost, i.e. $D_q(\hat{y}(i)) = \max_{p\in\mathcal{P}_k} D_p(\hat{y}(i))$, and decrease its splitting ratio as follows: $\phi_q(i) = \phi_q(i-1) - (1/g)\phi_q(i-1)$.
3. Choose another path $r \in \mathcal{P}_k$ either
   a) randomly, or
   b) so that the path cost is minimized, i.e. $D_r(\hat{y}(i)) = \min_{p\in\mathcal{P}_k} D_p(\hat{y}(i))$,

   and increase its splitting ratio as follows: $\phi_r(i) = \phi_r(i-1) + (1/g)\phi_q(i-1)$.
4. For all other paths $p \in \mathcal{P}_k$, keep the old splitting ratios.

5 NUMERICAL EVALUATION OF THE ALGORITHM

In this section we evaluate the performance of the proposed algorithm. First, in Subsection 5.1, a simple but efficient numerical evaluation method is developed. Thereafter, in Subsection 5.2, the results of this evaluation method applied to three different test networks are presented. Both the minimization of the maximum utilization and the minimization of the mean delay are addressed.

5.1 Evaluation method

The evaluation method is iterative and runs as follows. The test network (including the nodes n, links l, IE-pairs k, and paths p) and the traffic demands d_k are first fixed.[3] Split-

[3]Note that the traffic demands are used only for the *evaluation* of the proposed load balancing algorithm. The algorithm itself does *not* use any information on these demands.

ting ratios ϕ_p are initiated by allocating the traffic demands to the paths with the minimum hop-count. At each iteration i, the link loads $y_l(i)$ induced by the splitting ratios $\phi_p(i-1)$ are calculated by

$$y_l(i) = \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k : l \in p} d_k \phi_p(i-1).$$

The effect of traffic fluctuations on measured link loads is first ignored by assuming that measured link loads correspond to these induced values, $\hat{y}_l(i) = y_l(i)$. The new splitting ratios $\phi_p(i)$ are determined from these measured loads $\hat{y}_l(i)$ as presented in the previous section.

5.2 Numerical results

Three different test networks were used in the evaluation of the algorithm. The first one consists of 10 nodes, 52 links, and 72 IE-pairs, the second one of 15 nodes, 56 links, and 4 IE-pairs and the third one of 20 nodes, 102 links, and 380 IE-pairs. The first and third networks are random networks generated by a mechanism called MPLS Optimized Multipath [6]. The second one was taken from [7]. The number of possible paths, which may be huge, was limited by requiring that the paths $p \in \mathcal{P}_k$ do not have any common links. This policy results in a reasonable number of paths and provides the greatest diversity of links.

Minimization of the maximum link utilization. First we present the results related to the minimization of the maximum link utilization with cost function (4). The following variants of our adaptive load balancing algorithm were studied: path cost (7) and rule 3a in the algorithm (`RNDPTH`), and path cost (7) and rule 3b in the algorithm (`MINCST`).

In the following figures the maximum link utilization is depicted as a function of the number of iterations i for different levels of granularity g. The results of the adaptive algorithm are compared to the optimal value (lower bound) and to the maximum link utilization resulting in the minimum hop-count routing (upper bound). The optimal value is obtained from the static load balancing problem (3) by writing a corresponding LP-formulation of the problem and solving the problem using Minos 5 solver module. The maximum link utilization resulting in the minimum hop-count routing is taken from the first iteration round of the algorithm.

Figure 1 shows the results related to the 10-node test network. The left hand side corresponds to the variant `RNDPTH` of the algorithm and the right hand side to `MINCST`. The traffic demands are such that the min-hop routing results in the maximum link utilization of 0.88 while the optimal value is 0.54. The values obtained from the adaptive algorithm after 100 iterations are close to optimal ones. With a coarse granularity ($g = 20$), the convergence happens naturally faster (about in 30 iterations) but oscillations are outstanding. With finer granularities ($g = 50, 100$), oscillations almost disappear. The two variants of the adaptive load balancing algorithm do not differ from each other.

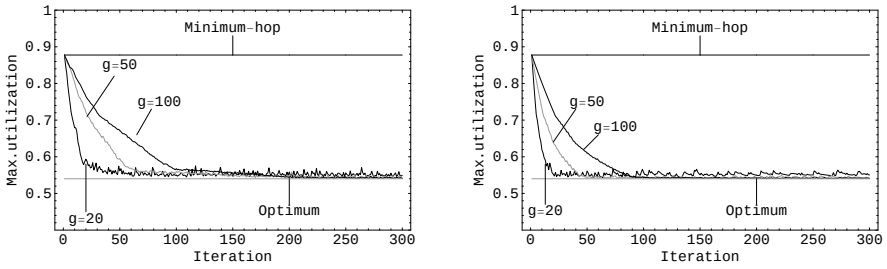

Figure 1: *The maximum link utilization in the 10-node network as a function of the number of iterations for different levels of granularity g. Left-hand-side: variant* RNDPTH*. Right-hand-side: variant* MINCST*.*

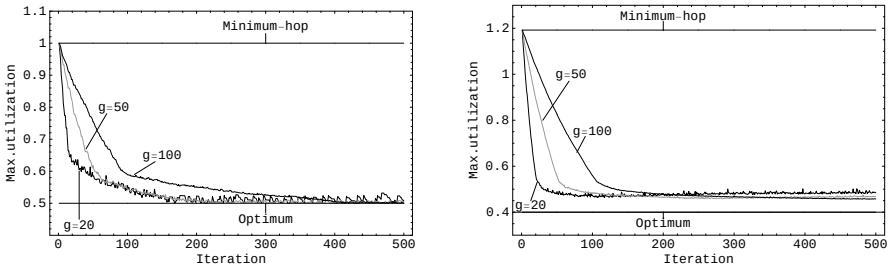

Figure 2: *The maximum link utilization of variant* RNDPTH *as a function of the number of iterations for different levels of granularity g. Left-hand-side: 15-node network. Right-hand-side: 20-node network.*

The results related to the RNDPTH variant applied to the test networks with 15 and 20 nodes are shown in Figure 2. The convergence times are only two times greater than in the 10-node network in spite of the huge growth in the complexity of the network. In the case of 20-node network there remains a gap between the steady state and the optimal values. However, the improvement to the performance of the minimum hop-count routing is outstanding.

Minimization of the total mean delay. Now we present the results related to the minimization of the total mean delay with cost function (2). The following variants of our adaptive load balancing algorithm were studied: path cost (5) and rule 3a in the algorithm (RNDPTH), path cost (5) and rule 3b in the algorithm (MINCST), and path cost (6) and rule 3b in the algorithm (MINDRV). For comparison, we also studied the performance of the

gradient-projection (G-P) algorithm for the multi-commodity flow problem as presented in [8].

Figure 3 shows the results related to 10-node test network with a coarse granularity, $g = 10$. The traffic demands are such that the min-hop routing results in the mean delay of 0.000179 while the optimal value is 0.000122. The optimal value is obtained from the static load balancing problem (1) by writing a corresponding NLP-program and solving it by Minos 5 solver module. The total mean delay resulting in the minimum hop-count routing is calculated from the link load values after the first iteration round of the algorithm. The convergence of the algorithm is achieved approximately in 20 iterations in all cases. We note that the performance of the simple algorithm does not differ significantly from that of the G-P algorithm. However, the number of updates per iteration in our adaptive algorithm is much smaller, since the splitting ratios of only two LSPs for each IE-pair are updated whereas in the G-P algorithm all the splitting ratios are updated.

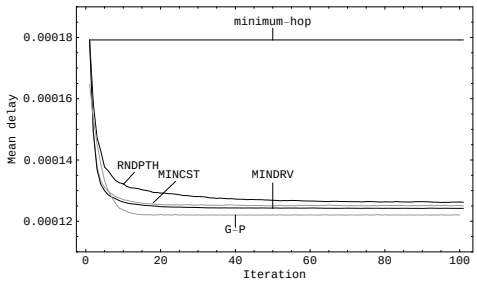

Figure 3: *The total mean delay in the 10-node network as a function of the number of iterations for the variants* RNDPTH, MINCST, *and* MINDRV *of the adaptive load balancing algorithm with granularity* $g = 10$ *together with the* G-P *algorithm.*

We have also studied the effect of random traffic fluctuation in the estimation of link loads. The preliminary results showed that errors in the measured link loads induce small oscillation to the network conditions but this can be reduced by the proper selection of the granularity.

6 CONCLUSIONS

In this paper we first reviewed some adaptive methods to balance the load in a MPLS network. Then we developed our own adaptive and distributed algorithm for the purpose using a simple heuristic approach. The idea is to measure the link loads periodically and then gradually move traffic from the congested part of the network to the less congested part. We considered two objectives, the minimization of the maximum link utilization and that of the total mean delay. A numerical evaluation method was also developed to study the performance of the algorithm. The performance was evaluated in three test networks.

The obtained steady state results were in many cases very close to the optimal values and in all cases much better than those of the standard min-hop routing. When the maximum link utilization was minimized, the variants `RNDPTH` and `MINCST` give similar results. The mean delay of the network is minimized most effectively by the variant `MINDRV` of algorithm moving traffic from the path with the maximum first derivative length to the path with the minimum first derivative length. The convergence time to obtain steady state were reasonable in all cases, even when we increased the complexity of the network substantially.

Acknowledgements. This work was financially supported by the Academy of Finland (grant n:o 74524).

BIBLIOGRAPHY

[1] E. Rosen, A. Viswanathan and R. Callon, *Multiprotocol Label Switching Architecture* IETF RFC3031, January 2001.

[2] D. Bertsekas and R. Gallager, Data Networks, Prentice-Hall, 2nd ed., 1992.

[3] A. Elwalid, C. Jin, S. Low and I. Widjaja, *MATE: MPLS Adaptive Traffic Engineering*, IEEE Infocom 2001.

[4] J. Song, S. Kim, M. Lee, H. Lee and T. Suda, Adaptive Load Distribution over Multipath in MPLS Networks, ICC'03, Anchorage, Alaska, May 2003.

[5] S. Butenweg, *Two distributed reactive MPLS Traffic Engineering mechanisms for throughput optimization in Best Effort MPLS networks*, IEEE Symposium on Computers and Communications, July 2003.

[6] C. Villamizar, *MPLS Optimized Multipath (MPLS-OMP)*, Internet draft <draft-villamizar-mpls-omp-01>, February 1999.

[7] M. Kodialam and T.V. Lakshman, *Dynamic Routing of Bandwidth Guarantees Tunnels with Restoration*, IEEE Infocom 2000.

[8] J. Tsitsiklis and D. Bertsekas, *Distributed Asynchronous Optimal Routing in Data Networks*, IEEE Transactions on Automatic Control, Vol. AC-31, No. 4, April 1986.