Adaptive load balancing with OSPF

Riikka Susitaival and Samuli Aalto
Networking Laboratory, Helsinki University of Technology
Email: {riikka.susitaival, samuli.aalto}@hut.fi

Abstract

The objective of load balancing is to move traffic from congested links to other parts of the network. If the traffic demands are known, load balancing can be formulated as an optimization problem. The resulting traffic allocation can be realized in the networks that use explicit routes, such as MPLS-networks. It has recently been found that similar load balancing is possible to be implemented even in the IP networks based on OSPF-routing by adjusting OSPF-weights of the links and traffic splitting ratios in the routers. However, if the traffic demands are unknown or they may change rapidly, another approach is needed. In this paper we study adaptive load balancing in OSPF-networks based on measured link loads. We propose an adaptive and distributed algorithm that gradually balances the load by making small changes in the traffic splitting ratios in the routers. We develope different traffic scenarios for testing the algorithm numerically. The results show that the performance of OSPF-networks can significantly be improved by our simple algorithm as compared to the equal splitting.
Keywords: OSPF, Traffic Engineering, adaptive routing, load balancing

1 Introduction

Traditionally traffic is routed along the minimum-hop paths in IP networks. By this approach the usage of link resources is minimized. However, some links may become congested while others remain underloaded. Recent studies have shown that although most of the links of networks are clearly under-utilized, the load of a link can be over 90% during traffic spikes [1]. In addition, many emerging applications, such as peer-to-peer applications, are bandwidth intensive in their nature. Thus efficient use of network resources necessitates implementation of some of traffic engineering mechanisms.

The idea of traffic-aware routing, and specially load balancing, is to avoid congested links when traffic is routed from a router to another. There are two distinct methodologies to implement traffic-aware routing in IP networks. The first one uses current routing protocols like Open Shortest Path First (OSPF) [2] but the link weights and the traffic splitting ratios in the routers are defined differently from the traditional approach. The second one takes advantage of some explicit routing protocol like Multi Protocol Label Switching (MPLS) [3] and defines the used paths beforehand.

One of the first proposals to tune OSPF-weights to achieve an optimal load distribution is presented by Fortz and Thorup in [4]. They assume that the routers are bound to split the traffic to a fixed destination equally to the admissible[1] next hops. Under this assumption, however, it is not possible to select the OSPF-weights so that the load distribution is optimal. Even finding the best possible weight setting is shown to be a NP-hard problem. Instead, Fortz and Thorup propose a local heuristic search method for setting the OSPF-weights. In [5] the same authors point out that OSPF-weight changes should be avoided as much as possible, because they confuse the active routing and the performance of TCP goes down.

The load balancing problem in IP networks based on OSPF-routing (OSPF-networks) is also investigated by Wang et al. in [6]. They show that optimal routing in terms of any objective function can be converted to a shortest-path routing with positive link weights if the traffic of each ingress-egress pair can be split arbitrarily to the shortest paths. However, in current IP routing with OSPF, only equal splitting is possible and, furthermore, the splitting in each router is done based on the destination address only.

Sridharan et al. [7] solve the problems appeared in [6]. The source-destination based splitting is easy to convert to destination based splitting by dividing the sums of incoming and outgoing traffic at the node. The problem of the unequal splitting ratios is solved by taking advantage of the existence of multiple prefixes to a certain destination. For a particular prefix only part of next hops are available. As the size of the routing table increases this approach approximates well the arbitrary splitting ratios of the optimal routing.

The problem of the approaches presented above is that the traffic demands are assumed to be known. Defining the traffic demand matrix describing the point-to-point demnads in the network is not straightforward task. If the traffic demands are not known, or the traffic conditions may change unexpectedly, another approach is needed. One possibility is to adaptively react to changes in the traffic detected by measurements, such as end-to-end monitoring or monitoring of each link individually. One of mechanisms to to improve the performance of the network adaptively uses a bandwidth estimation method to compute new link weights to be distributed [8] and another attempts to minimize the packet loss rate by optimizing the OSPF weights by online simulation [9]. However, it is not desirable to modify the link weights too frequently, since this can lead to undesired effects in the network performance and in addition, the optimization of link weights requires lots of computation.

Some adaptive mechanisms without link weight optimization developed recently are presented in

[1] A next hop is defined here to be admissible if it belongs to some shortest path to the destination.

[10] and [11]. In OSPF Optimized Multipath (OSPF-OMP) [10] load is balanced by moving traffic from the paths that include critically loaded segments to other paths according to ”move increments”, which are defined dynamically. As OSPF-OMP, Adaptive Multi-Path (AMP) establishes multiple paths to each destination by allowing also the use of non-shortest paths and balances load by changing the splitting ratios at each router. However, in AMP a node has only local information of its adjacent links. In paper [11] it remains unclear how much the use of only local information affects the performance of AMP. Another problem is that the studies above do not provide a proper comparison of the algorithms to minimum-hop routing and optimal routing, and the analysis of the stability issues is missing. Thus development of adaptive load balancing in OSPF networks is required.

In paper [12] we studied adaptive load balancing in MPLS-networks. In the present paper we study how similar ideas can be applied in OSPF-networks. Our assumption is that the link loads are measured periodically and the information on the measured loads is distributed to all routers. We suggest an adaptive and distributed algorithm to improve the performance of the network without knowledge of the traffic demands. The idea is that, based on the measured link loads, the routers make independently small changes in the load distribution by adjusting their own traffic splitting ratios. The algorithm is simple and does not need any special protocol to distribute measurements. We develop a numerical evaluation method and a measurement based traffic demand model to test functioning of the algorithm properly.

The rest of the paper is organized as follows. In section 2 we first review a static load balancing problem for off-line optimization of OSPF-weights and then formulate another optimization problem for adjusting the splitting ratios when the paths are fixed. The adaptive and distributed algorithm to optimize the splitting ratios is presented in section 3, and the performance of the proposed algorithm in different test networks and under various traffic conditions is evaluated numerically in section 4. Section 5 concludes the paper.

2 Load balancing based on known traffic demands

In this section we consider a static load balancing problem, in which the traffic demands are assumed to be known. We start with an OSPF-network model. Then we consider the optimization problem in general, after which we review how the OSPF-weights can be determined so that the optimal performance is achieved using shortest path routing. Finally we consider the case where the paths are fixed and only the traffic splitting ratios in the routers may be optimized.

2.1 Network model

Consider an IP network based on OSPF-routing (OSPF-network). Let $\mathcal{N}$ denote the set of nodes (routers) n and $\mathcal{L}$ the set of links l of the network. Alternatively we use notation (i, j) for a link from node i to node j. The capacity of link l is denoted by b_l. The set of ingress-egress (IE) pairs

$k = (s_k, t_k)$ is denoted by $\mathcal{K}$ with s_k referring to the ingress node and t_k referring to the egress node of IE-pair k. Let $\mathcal{P}_k$ denote the set of all possible paths p from node s_k to node t_k. We use notation $l \in p$ if link l belongs to path p. The traffic demand of IE-pair k is denoted by d_k.

In the link state based routing protocols like OSPF, each link l is associated with a fixed weight w_l and the traffic is carried along shortest paths. Let $\mathcal{P}_k^{\text{SP}}$ denote the set of shortest paths from node s_k to node t_k with respect to link weights w_l,

$$\mathcal{P}_k^{\text{SP}} = \{p \in \mathcal{P}_k \mid \sum_{l \in p} w_l = \min_{p' \in \mathcal{P}_k} \sum_{l' \in p'} w_{l'}\}.$$

The standard choice $w_l = 1$ for all l results in minimum-hop paths and, thus, minimizes the total required link bandwidth.

In each node i, the incoming traffic with the same destination t is aggregated and then split to links (i, j) that belong to some shortest path of the ingress-egress pair (i, t). Such adjacent nodes j are called admissible next hops. Let ϕ_{ij}^t denote the corresponding splitting ratios. Thus, ϕ_{ij}^t refers to the fraction of overall traffic passing node i and destined to node t that is forwarded on link (i, j). It is required that

$$\sum_{j:(i,j)\in p \text{ for some } p\in\mathcal{P}_{(i,t)}^{\text{SP}}} \phi_{ij}^t = 1.$$

For an illustration, see Figure 1. As, e.g., in [4], it is usually assumed that these splitting ratios ϕ_{ij}^t are equal,

$$\phi_{ij}^t = \frac{1}{|\{j' : (i, j') \in p \text{ for some } p \in \mathcal{P}_{(i,t)}^{\text{SP}}\}|}.$$

This choice is referred to as Equal Cost Multiple Path (ECMP). However, as mentioned in section 1, there is a method that allows unequal splitting ratios [7]. It may happen that the load for the bottleneck

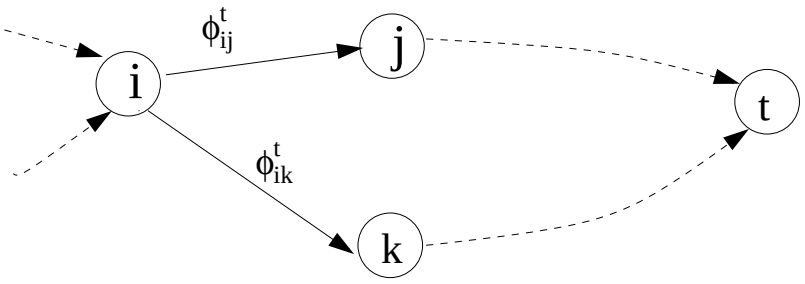

Figure 1: The network model

links becomes too high. Then, one naturally asks whether it is possible to choose the link weights w_l and the splitting ratios ϕ_{ij}^t in such a clever way that the maximum link utilization is minimized. This question is answered in the following subsections.

2.2 Static load balancing problem

Load balancing[2] can be solved in terms of various objective functions. Minimizing the maximum link utilization and minimizing the mean delay of the network are two examples of them. We note that the first one may result in long paths consuming much resources, while the latter one both balances the load and gives preference to short paths. However, since minimizing the maximum link utilization can be formulated as linear programming problem, we concentrate on that in this paper.

The optimal solution to the minimization problem of the maximum link utilization is not unique in general. Among the optimal solutions, the one that minimizes the overall usage of the resources is the most reasonable. Thus it is convenient to formulate an Linear Programming -problem (LP-problem) that minimizes the maximum link utilization with a greater weight but also takes into account the overall usage of the resources with a smaller weight as, e.g., in [6]:

$$\begin{array}{ll} \text{Minimize } \alpha + r \displaystyle\sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}} x_l^k & \text{subject to the constraints} \\ \alpha \geq 0,\ x_l^k \geq 0, & \text{for each } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\ \displaystyle\sum_{k\in\mathcal{K}} x_l^k \leq \alpha b_l, & \text{for each } l \in \mathcal{L}, \\ Ax^k = R^k, & \text{for each } k \in \mathcal{K}, \end{array} \tag{1}$$

where α and x_l^k are the free variables describing the minimum of the maximum link utilization and the traffic load of IE-pair k on link l, respectively, and r is some small constant. Furthermore, $A \in \mathbb{R}^{N\times L}$, where $N = |\mathcal{N}|$ and $L = |\mathcal{L}|$, denotes the matrix for which $A_{nl} = -1$ if link l directs to node n, $A_{nl} = 1$ if link l leaves from node n, and $A_{nl} = 0$ otherwise; $x^k \in \mathbb{R}^{L\times 1}$, $k \in \mathcal{K}$, refers to the link load vector with elements x_l^k; and $R^k \in \mathbb{R}^{N\times 1}$, $k \in \mathcal{K}$, denotes the vector for which $R_{s_k}^k = d_k$, $R_{t_k}^k = -d_k$, and $R_n^k = 0$ otherwise.

From the optimal traffic loads x_l^k it is possible to determine the set $\mathcal{P}_k^{\text{LB}}$ of paths p that are used to carry the traffic demand d_k from node s_k to node t_k,

$$\mathcal{P}_k^{\text{LB}} = \{p \in \mathcal{P}_k \mid x_l^k > 0 \text{ for all } l \in p\}.$$

2.3 Load balancing in OSPF-networks

Wang et al. [6] proved that there is a set of positive link weights w_l so that the optimal paths in the load balancing problem (1) are shortest paths with respect to these link weights. In other words, $\mathcal{P}_k^{\text{LB}} \subseteq \mathcal{P}_k^{\text{SP}}$ for all k. The procedure to define these link weights is given below.

Let $\tilde{y}_l = \sum_k \tilde{x}_l^k$ denote the traffic load allocated to link l in the optimal solution $\tilde{x}_l^k$ of the load balancing problem (1). Formulate then another LP-problem (primal) and its dual. In the primal LP-

[2] Also known as optimal routing.

problem the induced traffic loads $\tilde{y}_l$ serve as new capacity constraints:

$$\begin{array}{ll} \text{Minimize } \sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}} x_l^k & \text{subject to the constraints} \\ x_l^k \geq 0, & \text{for each } k \in \mathcal{K} \text{ and } l \in \mathcal{L}, \\ \sum_{k\in\mathcal{K}} x_l^k \leq \tilde{y}_l, & \text{for each } l \in \mathcal{L}\ , \\ Ax^k = R^k, & \text{for each } k \in \mathcal{K}, \end{array} \tag{2}$$

The dual of the problem above is:

$$\begin{array}{ll} \text{Maximize } \sum_{k\in\mathcal{K}} d_k U_{t_k}^k - \sum_{l\in\mathcal{L}} \tilde{y}_l W_l & \text{subject to the constraints} \\ W_l \geq 0 & \text{for each } l \in \mathcal{L}\ , \\ U_{s_k}^k = 0, & \text{for each } k \in \mathcal{K} \\ U_j^k - U_i^k \leq W_{(i,j)} + 1, & \text{for each } k \in \mathcal{K} \text{ and } (i,j) \in \mathcal{L}. \end{array} \tag{3}$$

The required link weights are then given by $w_l = W_l + 1$, where the variables W_l are determined as the solution to the dual problem.

In addition, optimal destination based traffic splitting ratios ϕ_{ij}^t are determined from the link loads x_l^k of the solution of the primal problem. These splitting ratios are calculated as follows [7]:

$$\phi_{ij}^t = \frac{\sum_{k:t_k=t} x_{(i,j)}^k}{\sum_{j':(i,j')\in\mathcal{L}} \sum_{k:t_k=t} x_{(i,j')}^k} \tag{4}$$

To summarize, the answer to the question posed at the end of subsection 2.1 is positive: it is possible to choose the link weights w_l and the traffic splitting ratios ϕ_{ij}^t in such a way that the maximum link utilization is minimized.

2.4 Optimization of the splitting ratios

The traffic control reacting to changes in traffic demands by changing the link weights is often too time consuming or impractical. In such a case with fixed link weights, we can still affect the traffic distribution by optimizing the traffic splitting ratios used in the routers.

We present a procedure to determine the splitting ratios that minimize the maximum link utilization with the given link weights. As before, let $\mathcal{P}_k^{\text{SP}}$ denote the set of shortest paths for IE-pair k with respect to these link weights. Let ϕ_p denote the fraction of traffic demand d_k that uses path $p \in \mathcal{P}_k^{\text{SP}}$.

We start by solving these splitting ratios for each IE-pair k from the following LP-problem:

$$\begin{aligned}
&\text{Minimize } \alpha + r \sum_{k\in\mathcal{K}} \sum_{l\in\mathcal{L}} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}: l\in p} d_k \phi_p && \text{subject to the constraints} \\
&\alpha \geq 0,\ \phi_p \geq 0, && \text{for each } p \in \bigcup_{k\in\mathcal{K}} \mathcal{P}_k^{\mathrm{SP}}, \\
&\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}: l\in p} d_k \phi_p \leq \alpha b_l, && \text{for each } l \in \mathcal{L}, \\
&\sum_{p\in\mathcal{P}_k} \phi_p = 1, && \text{for each } k \in \mathcal{K}.
\end{aligned} \tag{5}$$

Let ϕ_p be the optimal traffic share on path p. This induces the following link loads:

$$x_l^k = \sum_{p\in\mathcal{P}_k^{\mathrm{SP}}: l\in p} d_k \phi_p.$$

The destination based splitting ratios for each node i can then be calculated as in (4).

3 Adaptive load balancing

The static load balancing problem presented in the previous section is possible to be formulated and solved only if the traffic demands d_k are known. 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 this section we first formulate the corresponding dynamic load balancing problem for OSPF-networks and then describe an adaptive and distributed algorithm to solve the dynamic problem.

3.1 Dynamic load balancing problem

Our assumptions are as follows. The traffic demands of the network are unknown. The link loads are periodically measured at times t_n at each node adjacent to the link. Let $\hat{y}_l(n)$ denote the measured link load of link l from the measurement period (t_{n-1}, t_n). The information on the measured loads is distributed to all nodes in the network. This is done by flooding mechanism of OSPF routing protocol and will be described in more detail later. The time needed to distribute the information is negligible in comparison to the length of the measurement period.

In general, the objective of our dynamic load balancing problem is as follows. Based on the measured link loads, the link weights w_l and the traffic splitting ratios ϕ_{ij}^t should be adjusted so that they converge, as soon as possible, to the (unknown) optimal values of the corresponding static load balancing problem presented in subsection 2.3.

However, as mentioned in section 1, it is not desirable to modify the link weights too frequently. Therefore, we consider the dynamic problem at two time scales. At the shorter time scale, only the traffic splitting ratios are adjusted but the link weights are kept fixed. The objective in this case

is to adjust the traffic splitting ratios so that they converge to the (unknown) optimal values of the corresponding restricted optimization problem presented in subsection 2.4. At the longer time scale, also the link weights should be adjusted so that the optimal load distribution is finally achieved. One way to do it is to estimate the traffic demands from the measurement data and then determine the link weights as a solution to the dual problem presented in subsection 2.3.

In this paper we focus on dynamic load balancing at the shorter time scale. An destination-based, adaptive and distributed algorithm to solve this problem is described in the following subsection.

3.2 Adaptive and distributed algorithm for load balancing

We assume that the link weights w_l are fixed. For each IE-pair k, let $\mathcal{P}_k^{\mathrm{SP}}$ denote the set of shortest paths from node s_k to node t_k with respect to these link weights w_l.

Let $\phi_{ij}^t(n)$ denote the traffic splitting ratios that are based on the measured link loads $\hat{y}(n) = (\hat{y}_l(n); l \in \mathcal{L})$. We note that, since the measured link loads are distributed to all nodes, the decisions concerning the traffic splitting ratios can be done in a distributed way. Thus, in our adaptive and distributed algorithm, each node i independently determines the traffic splitting ratios ϕ_{ij}^t for all destination nodes t and admissible next hops j.

The decisions in the algorithm are based on a cost function $D_p(y)$ defined for each path $p \in \mathcal{P}_{(i,t)}^{\mathrm{SP}}$ by

$$D_p(y) = \max_{l \in p} \frac{y_l}{b_l},$$

where $y = (y_l; l \in \mathcal{L})$. This is a natural choice as the objective is to minimize the maximum link utilization. The idea in the algorithm is simply to alleviate the congestion on the most costly path by reducing the corresponding traffic splitting ratio. This should, of course, be compensated by increasing the splitting ratio related to some other path. A problem in adaptive adjustment of the splitting ratios at a short time scale is the possible disorder of the packets. However, this can be solved by changing only a part of the splitting ratios at a time, for example.

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. A finer granularity is achieved by increasing the value of g. The measurement period should be short enough to obtain reasonably fast convergence.

Algorithm At time t_n, after receiving the information $\hat{y}(n)$ concerning all the measured loads, node i adjusts the traffic splitting ratios for all destination nodes t as follows:

1. Calculate the cost $D_p(\hat{y}(n))$ for each path $p \in \mathcal{P}_{(i,t)}^{\mathrm{SP}}$.

2. Find the path $q \in \mathcal{P}^{\text{SP}}_{(i,t)}$ with maximum cost, i.e. $D_q(\hat{y}(n)) = \max_{p \in \mathcal{P}^{\text{SP}}_{(i,t)}} D_p(\hat{y}(n))$, and decrease the splitting ratio of the first link (i,j) of that path as follows:

$$\phi^t_{ij}(n) = \phi^t_{ij}(n-1) - \frac{1}{g}\phi^t_{ij}(n-1).$$

3. Choose another path $r \in \mathcal{P}^{\text{SP}}_{(i,t)}$ randomly and increase the splitting ratio of its first link (i,k) as follows:

$$\phi^t_{ik}(n) = \phi^t_{ik}(n-1) + \frac{1}{g}\phi^t_{ij}(n-1).$$

4. For all other admissible next hops j', keep the old splitting ratio,

$$\phi^t_{ij'}(n) = \phi^t_{ij'}(n-1).$$

On computational complexity Important question of the proposed algorithm is its computational complexity, since the feasibility and scalability of the algorithm depends greatly on that feature. In the algorithm each node adjusts its routing parameters in a distributed manner. Thus it is sufficient to estimate the algorithmic complexity at each node separately.

Let $|\mathcal{N}|$ be the size of the network. Given node i proceeds the steps of the algorithm for all other $|\mathcal{N}| - 1$ destination nodes. Let P be the number of parallel shortest paths and L be the length of a path. In the first step of the algorithm, the path cost, which includes L sum operations, is calculated for P paths. Thus the number of the operations in the first step is $L \times P$ per each destination.

In the second step the path with maximum cost is selected. The calculation can actually be done in the first step already, if during path cost calculation the momentary maximum of the path costs is stored in memory. Decreasing of splitting ratio in the second step is a standard time operation as well as choosing a random path and decreasing a splitting ratio of that in the third step. Finally, the fourth step does not need calculation at all.

In total, each iteration round of the algorithm at each node requires $|\mathcal{N}| - 1 \times P \times L$ calculations. OSPF specification [2] does not limit the number of parallel paths, but in implementations of IP routers P is typically bounded from 4 to 8 for practical reasons. Path length L is depends on the size of the network. The worst case is the chain of nodes, where L equals to $|\mathcal{N}| - 1$. However, in random networks L is closer to $\log |\mathcal{N}|$. Thus, as the size of the network increases, the number of calculations $|\mathcal{N}| - 1 \times P \times L$ approaches $|\mathcal{N}| \log |\mathcal{N}|$.

3.3 Flooding measurement information

In OSPF routing the routers exchange information about the link states and the metrics [2]. The flooding protocol is activated when the link state changes occur. The idea of the protocol in its simplicity is the following: Router A detects a change in the status of some of its adjacent links and updates the link state database accordingly. After that router A sends link state advertisements (LSA)

to its neighbors, which forward them again. The routing by the OSPF shortest path calculation does not provide sufficient capabilities to serve different types of traffic properly and to avoid congestion situations in the network. Thus in RFC 3630 [13] new LSAs, also called as traffic engineering LSAs, have deployed. These LSAs have a standard header but the payload enables transmission of additional information.

We use these traffic engineering LSAs to spread information of the measured link loads in our adaptive load balancing algorithm. Thus implementation any additional flooding protocol is avoided and signalling overhead of load balancing is negligible. Time needed to flood information is principally same as the end-to-end network delay, which is short as compared to the time scale of load balancing.

4 Numerical performance evaluation

In this section we evaluate numerically the performance of the proposed adaptive load balancing algorithm. First, in subsection 4.1, a simple but efficient numerical evaluation method is described. The method is similar to that developed in [12]. Then we propose traffic demand and link failure models to be used by the evaluation method in subsection 4.2 and 4.3. Finally, in subsection 4.4, the results of this evaluation method applied to two different test networks are presented.

4.1 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 link weights w_l. Let d_k denote the fixed traffic demand of IE-pair k and $d_k(n)$ time-dependent traffic demand of same IE-pair [3] Traffic of each IE-pair is initially allocated to the shortest paths with respect to the fixed link weights w_l. If multiple shortest paths exist, traffic is initially split equally in each node (ECMP).

At each iteration n, the measured link loads $\hat{y}_l(n)$ induced by the splitting ratios $\phi_{ij}^t(n-1)$ are calculated as follows. First we calculate, for each IE-pair k, the induced traffic splitting ratios $\phi_p(n-1)$ for each path $p \in \mathcal{P}_k^{\text{SP}}$ by

$$\phi_p(n-1) = \prod_{(i,j)\in p} \phi_{ij}^{t_k}(n-1).$$

Then the measured link loads $\hat{y}_l(n)$ are determined by

$$\hat{y}_l(n) = \sum_{k\in\mathcal{K}} d_k(n) \sum_{p\in\mathcal{P}_k^{\text{SP}}: l\in p} \phi_p(n-1),$$

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

After this, the new traffic splitting ratios $\phi_{ij}^t(n)$ are determined from the measured loads $\hat{y}_l(n)$ as presented in subsection 3.2.

4.2 Traffic demand models

We use two different approaches to generate traffic demands $d_k(n)$ to the network. First one is simple deterministic model, and second is Gaussian IID model. We assume that these models capture essential traffic characteristics of the current IP backbones and using more complex models such as autoregressive ARIMA process is not required at the time scale of traffic engineering.

Deterministic traffic In the deterministic traffic model the traffic demands $d_k(n)$ are constant over time, that is, $d_k(n) = d_k$. In this model random traffic fluctuations in the time scale of measurements are ignored. The model is rather simplified but gives a possibility to study the convergence of the algorithm well. The length of measurement period is not fixed to any specific time period.

Gaussian traffic with diurnal traffic pattern Many measurement studies, such as [14] and [15], have shown that traffic of IP backbone network has both clear diurnal traffic pattern and noisy fluctuations around the mean. Cao et al. [14] proposes a *moving IID Gaussian model* for IE-flows, in which flows of IE-pair k are considered as realizations of corresponding stochastic process $(X_k(n); n = 1, ..., N)$. The flow model consists of a deterministic demand $E[X_k(n)]$ and random fluctuating term $D[X_k(n)]Z_k(n)$:

$$X_k(n) = E[X_k(n)] + D[X_k(n)]Z_k(n), \tag{6}$$

where $Z_k(n)$ is referred to as *standardized residual*. The residuals are assumed to be independent Gaussian random variables with mean 0 and unit variance.

We apply the above Gaussian model to our study and compose a measurement-based model for the time-dependent traffic demands. Let $m(n)$ be a moving sample-average and $s(n)$ be a moving sample-standard-deviation of measured traffic trace $x(n)$. Time-dependent traffic demand $d_k(n)$ is derived from the measurements by assuming that variation coefficient $c = s(n)/m(n)$ remains constant for the original traffic trace and a derived traffic demand. Thus, traffic demand $d_k(n)$ of IE-pair k can be formulated as

$$d_k(n) = \big[\frac{m(n)}{\max_i m(i)} + \frac{s(n)}{m(n)} z_k(n)\big] d_k, \tag{7}$$

where $z_k(n)$ is standardized residual corresponding to $Z_k(n)$ of Cao's model. The measured traffic trace $x(n)$ from one day is obtained from our recent link measurement study [16]. The moving sample-average and standard-deviation using one hour averaging period of the trace are presented in Figure 2. We can see that the traffic trace is really unstationary and has diurnal variation.

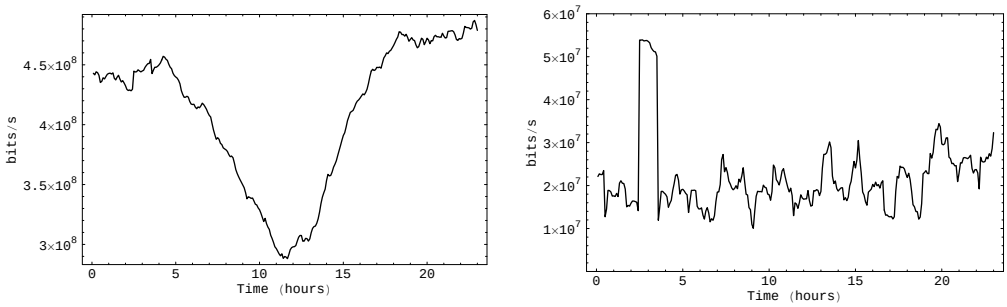

Figure 2: Left-hand-side: Standardized moving sample-average of a link measurement. Right-hand-side: standardized moving sample-standard-deviation of a link measurement.

The relevance of the moving IID Gaussian model is studied more in [16] and its follow-up study. To be short, the model might be valid for the traffic demands in IP backbone networks, if traffic is aggregated from sufficiently many sources and traffic volume is high.

4.3 Link failures

In addition to variation in traffic demands, the proposed load balancing algorithm should react also to unexpected changes in the traffic capacity. Such event could be a failure of a link, for example. According to the measurement study of Iannaccone et al. [17] about 50% of failures take less than one minute. Recovery from these short-term failures should be handled by other approaches as load balancing. However, same study shows that 20 % of link failures last over 10 minutes. In these long-term link failures the performance of the network can be improved by load balancing. From the measurements of [17] we conclude that median for time between link failures is 4 days. In our evaluation we use very simple link failure model, where each link failures during one day with probability $p = 0.1$. The time moment for the failure is random and duration is uniformly distributed between 10 and 100 minutes.

4.4 Numerical results

Two different test networks (see Figure 3) are used with the following characteristics:

1. 10 nodes, 52 links, and 72 IE-pairs;
2. 20 nodes, 102 links, and 380 IE-pairs.

From the test-networks we can define the nodes, the links between the nodes, IE-pairs and their deterministic traffic demands d_k. These fixed demands are then used to develop time-dependent traffic demands as explained above. The test networks are random networks generated by the mechanism described in [10].

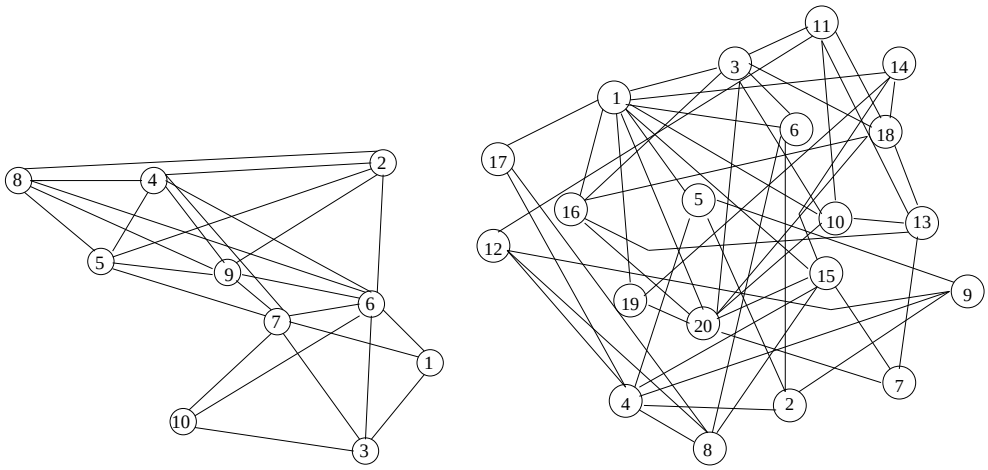

Figure 3: Left-hand-side: 10-node network. Right-hand-side: 20-node network.

The results of the adaptive algorithm are compared with the following approaches to forward traffic:

1. "Min-hop": traditional approach where traffic routed to a path that minimizes hop count.
2. "ECMP": the standard policy where the traffic is split equally to the shortest paths with the unit link weights.
3. "Sub-optimal": the optimal value of the restricted optimization problem (5) with link weights fixed to 1. This gives a lower bound to the load balancing algorithms that change only the splitting ratios but not the link weights.
4. "Optimal": the optimal value of the static load balancing problem (1). The lower bound for all heuristic load balancing algorithms, where both the splitting ratios and link weights can be adjusted.

A huge amount of computation effort is required to obtain optimal and sub-optimal reference values for every time moment when traffic demands or link capacities change. For this reason we have optimal values only for part of the test scenarios.

We study also a scenario, where load balancing is done at two time scales. Earlier we changed only the splitting ratios but now we assume that the link weights can be also optimized at a longer time scale (hours to days). At the longer time scale the optimal link weights and the splitting ratios corresponding to the original traffic demands d_k are determined from the dual problem (3) and the primal problem (2) of subsection 2.3, correspondingly. After that the link weights remain unchanged. The splitting ratios are thereafter balanced at the short time scale using our proposed algorithm. We refer this load balancing approach of two time scales by "OLW" ("Optimal Link Weights").

No traffic fluctuations In the first scenario our traffic model is deterministic. The shortest paths needed for the adaptive algorithm are calculated using link weights $w_l = 1$ for all links l. Note that in this scenario the length of measurement period is not fixed. We study the number of iterations required to convergence of the algorithm instead. The actual convergence time of the algorithm can be calculated by multiplying the number of iterations by the length of the measurement period. We have noted that the algorithm works well when granularity parameter is $g > 20$, which means that approximately 5% of the traffic load is moved at a time.

Figure 4 shows the resulting maximum link utilization for the 10-node and 20-node networks as a function of the number of iterations for granularity parameters $g = 20$ and $g = 50$. The adaptive algorithm is compared to minimum hop routing, ECMP, sub-optimal (unit link weights) and optimal results (also link weights optimized). We can see that in 10-node network the performance of the adaptive algorithm approaches the sub-optimal value and decreases the maximum link utilization remarkably as compared to the minimum hop routing and equal splitting. A small step size in the algorithm ensures that oscillations are insignificant. In 20-node network the optimal and sub-optimal results equals and thus optimal performance can be obtained by the adaptive algorithm.

The explanation for the differences in optimal and sub-optimal results of 10-node and 20-node networks is that, in the 10-node network, the number of shortest paths related to the unit link weights is 116 whereas it is 153 in the case of the optimal link weights. Thus the number of ϕ-parameters is greater in the latter case and also the results are better. In the 20-node network, the corresponding numbers of the shortest paths are 782 and 785 and the optimal and sub-optimal results are almost similar.

As conclusion, in both networks the adaptive algorithm converges to the lowest possible result. The convergence times are only two times greater in the 20-node network (approx. 200 iterations) than in the 10-node network (approx. 100 iterations) in spite of the huge growth in the complexity of the network.

Next we study the effect of link failures. In 10-node network 5 links failed at iterations 9, 66, 148, 228, and 241, correspondingly. The links were working again at iteration rounds 71, 135, 178, 247, and 264. The results are in the left side of Figure 5. We compare the adaptive algorithm (with $g = 20$) to minimum hop routing and ECMP. It can be seen that maximum link utilization changes dramatically in these events. In the iteration steps from 148 to 178 there is not enough capacity to carry traffic by minimum hop routing, since the maximum utilization of the links is over 1. By load balancing the maximum utilization remains under 80%. Results of load balancing in the 20-node network are in the right side of Figure 5. The number of the link failures is 10. Also in this case link failures affect the maximum link utilization much. Load balancing decreases the maximum link utilization approximately 55 % when compared to minimum-hop routing and 30 % when compared to ECMP.

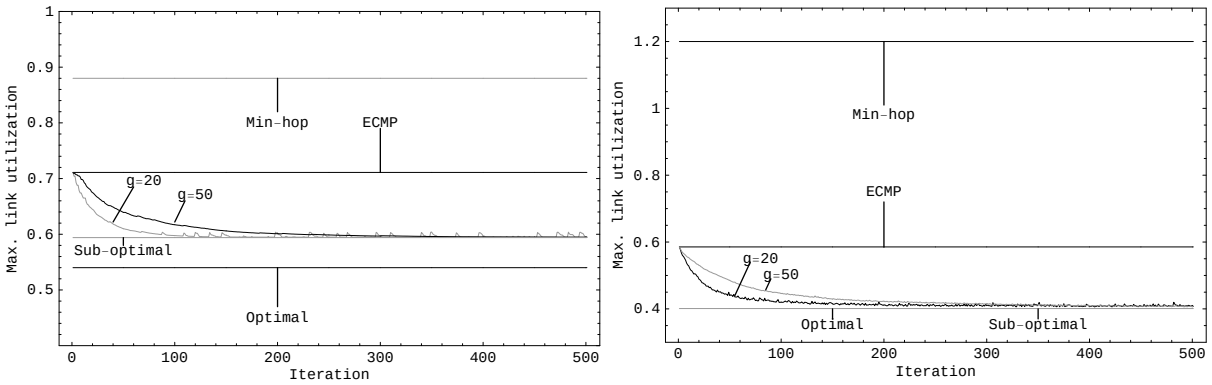

Figure 4: The maximum link utilization as a function of the number of iterations, when traffic demand is fixed. Left-hand-side: 10-node network. Right-hand-side: 20-node network.

Gaussian traffic and diurnal traffic pattern In this test scenario we use Gaussian IID traffic demand model. The model utilizes the measurements done in a backbone network and thus we are able to check whether the actual convergence time of the algorithm is short enough. In this case we have to also fix the measurement period. Selection of the length of the period and frequence to flood the measurement information is two-fold: Algorithm itself converges as faster as the measurements period is shorter. On the other hand, a short measurement period induces signaling overhead and oscillations to the performance of the algorithm. We compromise the period length on 5 minutes, which is also the length of SNMP measurements. The shortest paths needed for the adaptive algorithm are again calculated using link weights $w_l = 1$ for all links l.

Figure 6 shows the resulting maximum link utilization during one day for the 10-node and 20-node networks as a function of time for granularity parameters $g = 20$ and $g = 50$. In this case the adaptive algorithm is compared to minimum hop routing, ECMP, optimal (only in 10-node network) and Optimal Link Weight-approach (OLW). Again the proposed algorithm improves the performance of the network as compared to standard policies. The performance of the OLW (dashed line) is almost optimal. The initial convergence time of the adaptive algorithm is 2-3 hours. When the traffic demands change, the algorithm adapts to changes.

Next we combine link failures to Gaussian traffic model. The maximum link utilizations for the 10-node and 20-node networks as a function of time are presented in Figure 7. We compare adaptive algorithm ($g = 20$ and $g = 50$) with minimum hop routing, ECMP and OLW. In this scenario the network conditions change both slowly due to diurnal traffic pattern and dramatically due to link failures. Interesting is that although optimizing of the link weights by OLW strategy (dashed line) gives the best performance first, in the time period from 9th to 11th hour the performance of OLW is worse than that of the adaptive algorithm and even EMCP. The reason for poor performance of OLW is that link weights are far from optimal after the link failure. Thus using unit link weights can be

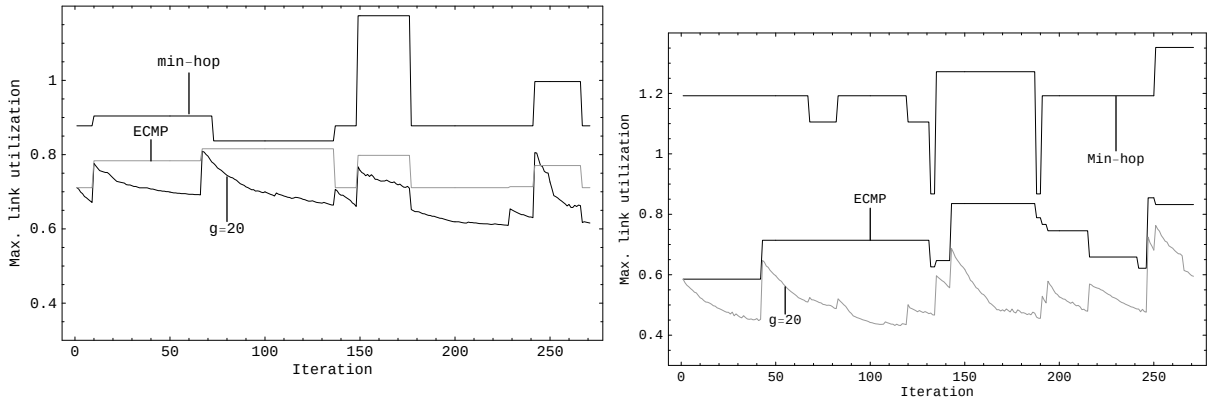

Figure 5: The maximum link utilization as a function of the number of iterations, when traffic demand is fixed and there are link failures. Left-hand-side: 10-node network. Right-hand-side: 20-node network.

better approach, if link capacities change much.

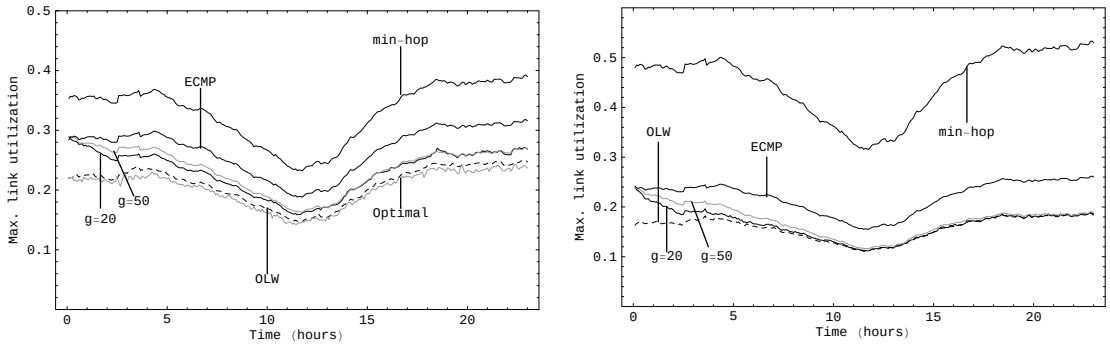

Figure 6: The maximum link utilization as a function of time, when traffic is Gaussian and has diurnal pattern. Left-hand-side: 10-node network. Right-hand-side: 20-node network.

Summary of results As a conclusion, the proposed adaptive algorithm improved the performance of the network in all test scenarios. When the network capacities changed dramatically, also the adaptive algorithm reacted. The optimization of the link weights at the longer time scale (OLW) improved the performance of the network little bit more but gave also poor performance in some cases. If 5 minutes measurement period is used, the convergence of the adaptive algorithm is short enough.

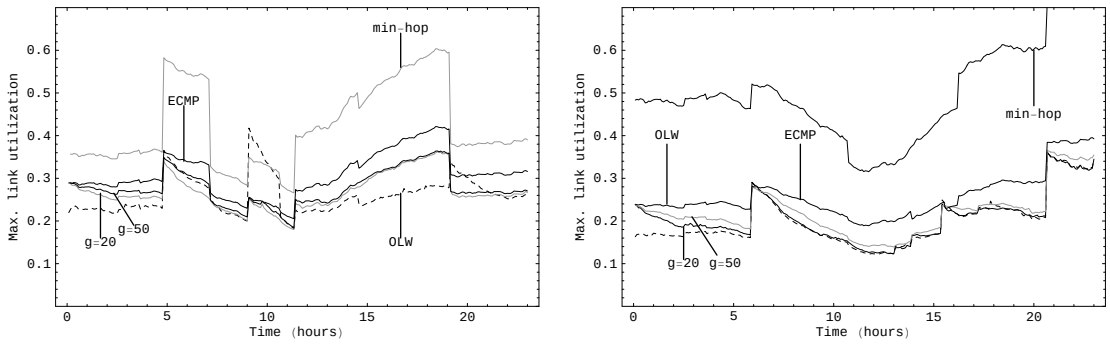

Figure 7: The maximum link utilization as a function of time, when traffic is Gaussian and has diurnal pattern and there are link failures. Left-hand-side: 10-node network. Right-hand-side: 20-node network.

5 Conclusion

In this paper we studied how load can be balanced adaptively in OSPF-networks using a distributed approach. We have proposed a simple adaptive algorithm for that and tested it in several traffic scenarios. The scenarios included both unstationary traffic demands and link failures. By adaptive algorithm it is possible to decrease the load of the network significantly and thus improve the performance of the network.

We also considered the procedure of optimizing the OSPF-weights by primal-dual methods and how this can be combined to adaptive heuristics. The results show that the optimization of traffic splitting ratios improves the performance of the network when compared to equal splitting. However, when network conditions change much, optimized link weights can produce worse performance than using unit weights.

In the future the approach which combines the shorter and longer time scale optimization has to be developed further. In addition, we have to study if the disorder of packets is really a problem and how this problem can be solved.

Acknowledgements. This work was financially supported by the Academy of Finland (grant n:o 74524). We like to also thank anonymous reviewers for valuable comments of the paper.

References

[1] E. J. ANDERSON, T. E. ANDERSON, On the Stability of Adaptive Routing in the Presence of Congestion Control, in Proceedings of IEEE INFOCOM, 2003.

[2] J. MOY, OSPF Version 2, IETF RFC2328, April 1998.

[3] E. ROSEN, A. VISWANATHAN AND R. CALLON, Multiprotocol Label Switching Architecture IETF RFC3031, January 2001.

[4] B. FORTZ, M. THORUP, Internet Traffic Engineering by Optimizing OSPF Weights, in Proc. of IEEE Infocom 2000.

[5] B. FORTZ AND M. THORUP Optimizing OSPF/IS-IS Weights in a Changing World, IEEE Journal on Selected Areas in Communications, Vol. 20, No. 4, pp. 756-767, 2002.

[6] Y. WANG, Z. WANG AND L. ZHANG Internet Traffic Engineering without Full Mesh Overlaying, in Proc. of IEEE Infocom 2001.

[7] A. SRIDHARAN, R. GUÉRIN AND C. DIOT, Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF-IS-IS Networks, in Proc. of IEEE Infocom 2003.

[8] T. B. PEREIRA AND L. L. LING, Network Performance Analysis of and Adaptive OSPF Routing Strategy-Effective Bandwidth Estimation, in International Telecommunication Symposium ITS 2002, Brazil.

[9] H. T. KAUR, T. YE, S. KALYANARAMAN, K. S. VASTOLA, Minimizing Packet Loss by Optimizing OSPF Weights Using On-line Simulation, in Proceedings of the 11th International IEEE/ACM Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. MASCOTS-2003, 2003.

[10] C. VILLAMIZAR, OSPF Optimized Multipath (OSPF-OMP), IETF Internet-draft, draft-ietf-ospf-omp-02, February 1999.

[11] I. GOJMERAC, T. ZIEGLER, F. RICCIATO AND P. REICHL, Adaptive Multipath Routing for Dynamic Traffic Engineering, in Proceedings of IEEE Globecom, Nov. 2003.

[12] R. SUSITAIVAL, S. AALTO AND J. VIRTAMO, Adaptive load balancing using MPLS, in Proc. of MMB&PGTS, 2004.

[13] D. KATZ, K. KOMPELLA AND D. YEUNG, Traffic Engineering (TE) Extensions to OSPF Version 2, IETF RFC 3630, September 2003.

[14] J. CAO, D. DAVIS, S. V. WIEL, AND B. YU, Time-varying network tomography, Journal of the American Statistical Association, Vol. 95, pp. 1063–1075, 2000.

[15] M. ROUGHAN, A. GREENBERG, C. KALAMANEK, M. RUMCEWITCZ, J. YATES AND Y. ZHANG, Experience in Measuring Internet Backbone Traffic Variability: Model, Metrics, Measurements and Meaning, in Proc. of ITC 2003.

[16] I. JUVA, R. SUSITAIVAL, M. PEUHKURI, AND S. AALTO, Traffic characterization for traffic engineering purposes: Analysis of Funet data, in Proc. of NGI 2005, Rome, Italy, April 2005.

[17] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, C. Diot, Analysis of link failures in an IP backbone, ACM SIGCOMM IMW2002.