DIMACS Technical Report 2001-17
May 2001

How good can IP routing be?

by

Dean H. Lorenz[1] Ariel Orda[2]
Department of Electrical Engineering
Technion—Israel Institute of Technology
{deanh@tx,ariel@ee}.technion.ac.il

Danny Raz[3] Yuval Shavitt[4]
Bell Laboratories
Lucent Technologies
{raz,shavitt}@research.bell-labs.com

[1]Part of this work was done while visiting Bell Labs, Lucent Technologies, and was supported in part by DIMACS.
[2]Part of this work was done while visiting Bell Labs, Lucent Technologies.
[3]Permanent DIMACS member. Current address: Department of Computer Science, Technion—Israel Institute of Technology, `danny@cs.technion.ac.il`.
[4]Permanent DIMACS member. Current address: Department of Electrical Engineering – Systems, Tel-Aviv University, `shavitt@eng.tau.ac.il`.

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

In the traditional IP scheme, both the packet forwarding and the routing protocols are source invariant, i.e., their decisions depend on the destination IP address and not on the source address. Recent protocols, such as MPLS, as well as traditional circuit based protocols like PNNI allow routing decisions to depend on both the source and destination addresses. In fact, much of the theoretical work on routing assumes per-flow forwarding and routing, i.e., the forwarding decision is based on both the source and destination addresses.

The benefit of per-flow forwarding is well-accepted, so is the practical implications of its deployment. Nevertheless, no quantitative study has been carried on the performance differences between the two approaches.

This work aims at investigating the toll in terms of performance degradation that is incurred by source invariant schemes, as opposed to the per-flow alternatives. We show, both theoretically and by simulations, that source invariant routing can be significantly worse than per-flow routing. Realizing that static shortest path algorithms are not optimal even among the source invariant routing algorithms, we develop novel routing algorithms that are based on dynamic weights, and empirically study their performance in an Internet like environment. We demonstrate that these new algorithms perform significantly better than standard IP routing schemes.

1 Introduction

In the traditional IP scheme, both the packet forwarding [Pos81] and routing protocols (e.g. RIP [MS95, Hed88, Mal98] and OSPF [Moy95, Moy98]) are *source invariant* i.e., their decisions depend solely on the destination IP address and not on the source address. Recent protocols, such as MPLS [RVC01], as well as traditional circuit-based protocols [PNN96] allow routing and forwarding decisions to depend on both the source and destination addresses. In fact, much of the theoretical work on routing assumes *per-flow* forwarding and routing, i.e., these decisions are based on both the source and destination addresses. The benefit of per-flow forwarding is well accepted, as well as the practical complications of its deployment. Nevertheless, no quantitative study has been carried on the performance differences between the source-invariant and per-flow approaches.

This work aims at investigating the performance gap between source invariant and per-flow schemes. By employing both theoretical analysis and simulation experiments, we demonstrate that the toll incurred by (standard) source invariant schemes is significant. On the other hand, per-flow schemes impose complications that usually make their deployment practically impossible. In particular, any solution that requires to consider some quadratic number of source-destination pairs (rather than a linear number of destinations) is far from being scalable. Facing these gaps between the two basic schemes, in this study we propose a novel source invariant scheme. Our scheme exhibits a significantly improved performance over the standard source invariant scheme, and comes close to the performance of per-flow schemes; at the same time, it maintains the practical advantage of independence of source addresses.

While offering a dramatic improvement in terms of performance, our scheme does come at a price, namely requiring a higher degree of centralization. However, increased centralization is one of the processes that can be observed in the evolution of the Internet. Originally, a decentralized infrastructure was a main design principle. However, with the growing importance of the Internet to the economic and social infrastructures, there has been a growing emphasis on continuous operability and increased utilization. These goals often call for some degree of centralized management. Taking a closer look, any management scheme for the inter-domain level need to remain distributed, as at that level the Internet consists of a (large) collection of autonomous systems, each managed by a different entity; indeed, network operation at that level remained distributed, bounded only by (BGP) policy rules. However, the picture is completely different at the intra-domain level. Here, the emergence of many networks competing as service providers pushed for differentiation in the quality of operation and the ability to lower prices based on higher resource utilization. This fierce competition pushed network providers to a more centralized control and management of their (autonomous) infrastructures.

Previous generations of management systems aimed mainly at monitoring the network performance and health. However, driven by the above processes, the current trend in network management is towards a centralized control of the network to achieve higher utilization and better predictability of its behavior. In particular, the IPNC system [IPN] enables to set the OSPF "weights" in a centralized manner. Consequently, a considerable body of work has been carried on how to set these weights in a way that improves certain performance measures [MSZ97, FT00]. However, in this work we show that, theoretically, any routing algorithm based on static weights can perform as bad as (the worst case of) any source invariant scheme. Our proof includes OSPF routing where at any point the flow can be split evenly to several sub-branches towards the destination. Accordingly, our new scheme exploits a significant capability of centralized management stations at the intra-domain level, namely having information on current traffic statistics, and uses this information in order to compute the forwarding tables at the various routers.

Main contributions. We show that theoretically the gap in performance (defined either as the load on the most congested link or as the maximum flow the network can support) between IP routing and OSPF may be as bad as $\Theta(N)$, where N is the number of nodes in the network. We also show that OSPF is $\Theta(N)$ worse than per flow routing (MPLS). This means that although OSPF may perform much better than traditional IP, in some cases it exhibits no advantage over tradition IP routing.

We show that if we use shortest path routing, then any static weights assignment may be bad ($\Theta(N)$) even if we use per flow routing. Thus, we present a family of centralized algorithms that set forwarding tables in IP networks, based on dynamically changing weights. In all the algorithms the link weights are exponential (similar to [AAP93]) in the load on the link. The centralized algorithm input is the network topology and a flow demand matrix. In practice, the demand matrix can be based on long term traffic statistics. The algorithms are shown to perform much better than IP routing on different Internet-like topologies and demand matrices.

Organization. The rest of this paper is structured as follows. In the next section, we formally define the model and the different routing schemes we handle. In Section 3 we show that the optimal IP routing is $\mathcal{NP}$-hard. In Section 4 we present theoretical upper and lower bounds for the gaps between the different routing schemes. In Section 5 we present the algorithms and their performance study. Finally, we discuss related work and future research directions.

2 Model and Problem Formulation

The network is defined as a (possibly directed) graph $G(V, E)$, $|V| = n,\ |E| = m$. Denote by $\mathcal{N}_v$ the set of neighbors of a node v. Each link $e \in E$ has a capacity c_e, $c_e > 0$. A demand matrix, $\mathcal{D} = \{d_{i,j}\}$, defines the demand $d_{i,j}$ between each source i and destination j, i.e., the amount of (i, j)-*flow*.

A *routing assignment* is a function $\mathcal{R} : V^4 \rightarrow [0..1]$, such that $\phi_{u,v}(i, j)$ is the relative amount of (i, j)-flow that is routed from a node u to a neighbor v. Such a function must comply with:

1. $\forall u, i, j \in V :\ \sum_{v \in \mathcal{N}_u} \phi_{u,v}(i, j) = 1$
2. $\forall u, i, j, v \in V,\ v \notin \mathcal{N}_u :\ \phi_{u,v}(i, j) = 0.$

A routing assignment $\mathcal{R}$ is *source invariant* if $\forall u, v, i_1, i_2, j \in V :\ \phi_{u,v}(i_1, j) = \phi_{u,v}(i_2, j) \equiv \phi_{u,v}(j)$.

A *routing paradigm* is a set of rules that characterizes a class of routing assignments. We define the following routing paradigms:

Unrestricted Splitable Routing (US-R) The class of all routing assignments, i.e., flow can be split among the outgoing links arbitrarily.

Restricted Splitable Routing (RS-R) A class of routing assignments in which flow can be split over

(at most) a *predetermined number* L of outgoing links, i.e.: $\forall u, i, j \in V: \ |\{v|\phi_{u,v}(i,j) > 0\}| \leq L$.
Remark: A special case of RS-R is when $L = 1$, which is known as the *unsplitable flow problem*, and shall be referred here as the RS-R$_1$ paradigm.
Standard IP Forwarding (IP-R) The special case of source-invariant RS-R$_1$, i.e., $\forall u, j \in V, \ \exists v$ such that $\phi_{u,v}(j) = 1$.
OSPF routing (OSPF-R) A class of source invariant routing assignments that split flow evenly among (non-null) next hops, i.e.: $\forall u, j, v_1, v_2 \in V$: if $\phi_{u,v_1}(j) > 0$ and $\phi_{u,v_2}(j) > 0$ then $\phi_{u,v_1}(j) = \phi_{u,v_2}(j)$.

By definition, source invariant routing assignments do not differentiate among source nodes in terms of the routing variables $\phi_{u,v}(j)$; yet, this does not necessarily imply that the actual routing of packets is source invariant. Consider, for example, path caching techniques, which are commonly employed in IP routers and attempt to reduce the frequency of out-of-order arrivals at the destination. There, upon reception of a packet that belongs to a "new" (i.e., non-cached) flow (i.e., source-destination pair), a new entry in the cache is opened according to some routing decision, which in turn is governed by the ϕ variables. That is, the ϕ variables specify the relative amount of *cache entries* per destination that correspond to an outgoing link. As a result, packets belonging to the same flow are routed to the same (cached) outgoing link. Accordingly, we shall consider two cases of source invariant routing: in the first, *basic* case, routing decisions are made for each packet independently; in the second, *flow-cached* case, the same outgoing link must be used for traffic that originates at the same source node.

For a given network, the demand matrix and routing assignment define a unique vector of link flows. These, in turn, do not necessarily comply with the link capacity constraints. Accordingly, we investigate two different scenarios. In the first, capacities are considered to be "soft" constraints, which can be violated at a "cost"; accordingly, our aim is to identify a routing assignment which decreases the maximal violation across the network (a more precise definition follows). In the second scenario, capacities are "absolute" constraints, which cannot be violated. This implies that, for a given routing assignment, the actual input rates should be reduced beyond the values of the demand matrix, so as to comply with the capacity constraints, hence defining an *allocation* for the source-destination pairs. This can be performed in more than one way; for our purposes, we assume that there is some rule that uniquely identifies an allocation matrix for any given network, (original) demand matrix and routing assignment. For example, a well known such rule is that of *max-min fairness*. Accordingly, we denote by $\hat{\mathcal{D}} = \hat{\mathcal{D}}(G, \mathcal{D}, \mathcal{R})$ the allocation matrix that results from the application of that rule to the network G, demand matrix $\mathcal{D}$ and routing assignment $\mathcal{R}$. The *throughput* of an allocation matrix is the sum of its components.

We proceed to formulate the above in a more precise manner. Given a vector of link flows, the *link congestion factor* is the ratio between the flow routed over the link and its capacity; the *network congestion factor* is then the largest link congestion factor. For a network G, a pair of routing assignment $\mathcal{R}$ and demand matrix $\mathcal{D}$ are said to be *feasible* if the resulting network congestion factor is at most 1; we then say that $\mathcal{R}$ is feasible for $\mathcal{D}$ and that $\mathcal{D}$ is feasible for $\mathcal{R}$. We observe that, by definition, a routing assignment $\mathcal{R}$ and its corresponding allocation matrix $\hat{\mathcal{D}}(G, \mathcal{D}, \mathcal{R})$ are feasible.

We are now ready to define our optimization problems.
Problem Congestion Factor: Given a routing paradigm, a network $G(V, E)$ with link capacities

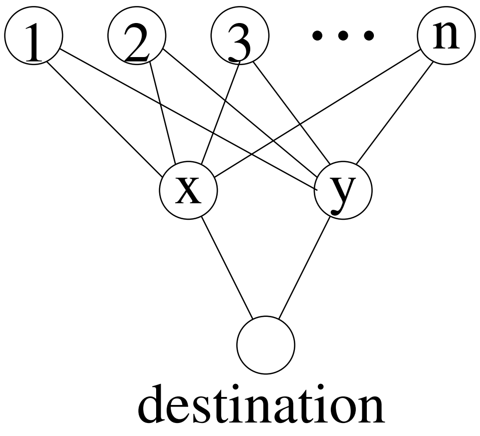

Figure 1: A reduction from the partition problem to optimal routing.

$\{c_e|e \in E\}$ and a demand matrix $\mathcal{D}$, find a routing assignment $\mathcal{R}$ that minimizes the network congestion factor.
Problem Max Flow: Given a routing paradigm, a network $G(V, E)$ with link capacities $\{c_e|e \in E\}$ and a demand matrix $\mathcal{D}$, find a routing assignment $\mathcal{R}$ such that the allocation matrix $\hat{\mathcal{D}}(G, \mathcal{D}, \mathcal{R})$ has maximum throughput.

3 Hardness Results

Next we show that finding an optimal IP routing (i.e., problem Congestion Factor with the routing paradigm IP-R) is $\mathcal{NP}$-hard even for a single destination. To that end, we prove that the *subset sum* problem [GJ79, Problem SP13] can be reduced to an optimal IP routing decision problem. The subset sum problem is defined as follows: given a_i, $i = 1, \ldots, n$ elements with sizes $s(a_i) \in Z^+$, and a positive integer B, find a subset of the elements whose size sum equals to B.

Theorem 1 *The decision optimal IP routing problem is at least as hard as the subset sum problem.*

Proof: We construct the following graph. For every element a_i create a node i with flow demand a_i to a destination d. Connect each node $i \in \{1, \ldots, n\}$ with two links of infinite capacity to nodes x and y (see Figure 1). Connect x and y to d with links of capacity $\max\{B,\ B - \sum_{i=1}^{n} a_i\}$. The partition can be made if the max load in the IP network can be smaller than 1. ■
Remark: The optimal flow routing problem is NP-hard as well, since in the network of Figure 1 the IP restriction does not affect the routing.
Remark: For a single destination without the IP restriction, some constant-factor approximations have been suggested [DGG99].

4 Theoretical Bounds

In this section we study the differences among the routing paradigms defined in Section 2. We show upper- and lower bounds on the worst case ratio between the performance of these paradigms.

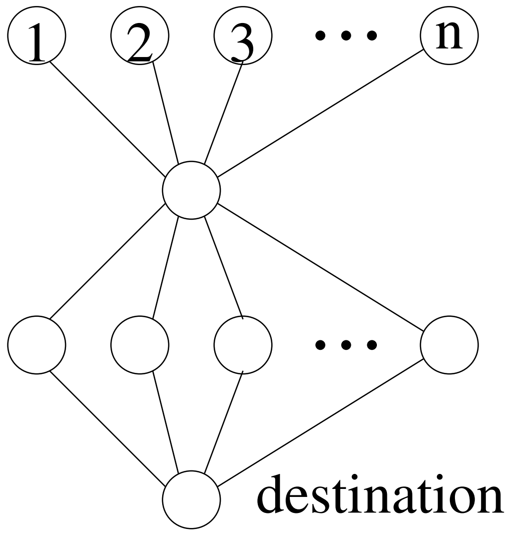

Figure 2: An example of the difference between IP routing and flow based routing

4.1 IP-R vs RS-R_1 and OSPF-R

We show that IP-R can be $\Omega(N)$ worse than RS-R_1 with respect to both optimization criteria.

Consider the example of Figure 2, where N sources are connected to a single destination over N link-disjoint all sharing an intermediate node. In IP-R, all the traffic is forced to use a single path from the shared intermediate node to the destination; in RS-R_1, each flow can take a separate route, and in OSPF-R the flows can be divided equally among the N links. Let all demand and link capacity values be equal to one. The network congestion factor is thus N for IP-R and 1 for RS-R_1 and OSPF-R, resulting in an $\Omega(N)$ factor. Similarly, the max flow is 1 in IP-R and N in RS-R_1 and OSPF-R, leading to the same $\Omega(N)$ factor.

We note that $O(N)$ is a straightforward upper bound. To realize that, consider first the case of a single destination. Given a routing assignment for RS-R_1, we construct the routing assignment for IP-R in the following way. Examine the source with the highest allocation under RS-R_1, and use its route for IP-R. Now, examine the other sources by decreasing order of allocation, and route them along the RS-R_1 route until they hit a node used by the routing. Obviously, the total allocated flow in IP-R is at least as large as the highest allocated flow under RS-R_1, hence it is at least $1/N$ of the total allocated flow under RS-R_1. Similarly, for the congestion factor, if a link is used in IP-R, at least $1/N$ of the allocation is used also under RS-R_1. Hence, we have established a $\Theta(N)$ factor for both criteria. In a similar way, when multiple destinations exist, the tight bound is the maximum number of sources per single destination (rather than their sum).

4.2 OSPF-R vs RS-R_1 under the Max Flow criterion: the flow-cached case

We turn to compare the performance of OSPF-R, in the flow-cached case, with that of RS-R_1, under the Max Flow criterion. Consider the network in Figure 3. $s_1, s_2, \cdots, s_N$ are the source nodes, each carrying a unit traffic demand to the common destination. The topology is composed of a cascade of $\log^* N$ identical components, each having $2N$ nodes. The link capacities are as depicted in the figure.

It is easy to verify that, under the RS-R$_1$ routing paradigm, all N units of demand can be shipped to their destination. Hence, the throughput is N. We proceed to upper-bound the throughput of the OSPF-R paradigm in the flow-cached case. Consider the first (uppermost) component in the cascade. At any of the $N-1$ nodes $u_1, \ldots, u_{N-1}$, the routing assignment can either direct all traffic to one link, or else split it between two. If only one link is chosen at all the $N-1$ nodes, then at most one unit of throughput can be achieved. Otherwise, suppose that u_i is the first node at which two links are chosen. By the OSPF-R paradigm, the same ϕ value, i.e., 0.5, is chosen for the two links. This means that the maximum amount of traffic that exists u_i is $\frac{N-i+1}{2} \leq \frac{N}{2}$.[1] To maximize the throughput one needs to split the routing as much as possible, hence, the maximum throughput out of the first cascaded component (hence, the maximum input to the second component) is $\log N$. Applying the same argument iteratively over all components, we conclude that the throughput out of the k-th component is at most $\log^{(k)} N$; as a result, the destination, which is located at the $\log^* N$-th component, receives at most 2 unit of throughput. Therefore, we have established an $\Omega(N)$ lower-bound for the ratio between the performance of RS-R$_1$ and OSPF-R under the Max Flow criterion in the flow-cached case.

4.3 OSPF-R vs RS-R$_1$ under the Max Flow criterion: the basic case

Consider again the network of Figure 3, however assume now that it has a single component, rather than $\log^* N$ in cascade. Clearly, the throughput of RS-R$_1$ is N here too. With OSPF-R, we note that any node that uses two outgoing links cannot exceed a throughput of 2; this is because in the basic case the equal-split rule applies at the *packet level*, hence the traffic on a link (u, v) is upper-bounded by the minimum capacity over all that emanate from u.[2] Hence, if any node uses more than one outgoing link, the throughput at the destination is at most 2; otherwise, i.e., only one link is used at all nodes, the throughput at the destination is 1. Hence, the $\Omega(N)$ lower-bound holds in the flow-cached case too.

4.4 OSPF-R vs RS-R$_1$ considering the Congestion Factor criterion

We turn to consider the Congestion Factor criterion. We recall that now all the input demand must be shipped, possibly creating congestion on links, i.e., an excess over capacity. As above, consider the first component in Figure 3. RS-R$_1$ can ship all the N units of demand with a congestion factor of one. OSPF-R[3] can either choose a single link at each node, hence resulting with a congestion factor of N at the last link, or else choose two links at (at least) one node. In the latter case, let u_i be the first such node; then, the flow over the single-capacity link emanating from u_i is $\frac{N}{2}$, hence the

[1] In practice, half of the flows are cached on the link with unit capacity, hence their total throughput in the respective $\hat{\mathcal{D}}$ allocation is 1.

[2] In practice, this capacity constraint is considered by the allocation rule whose outcome is the matrix $\hat{\mathcal{D}}$.

[3] As is easy to see, when considering the Congestion Factor criterion there is no need to distinguish between the basic and flow-cached cases.

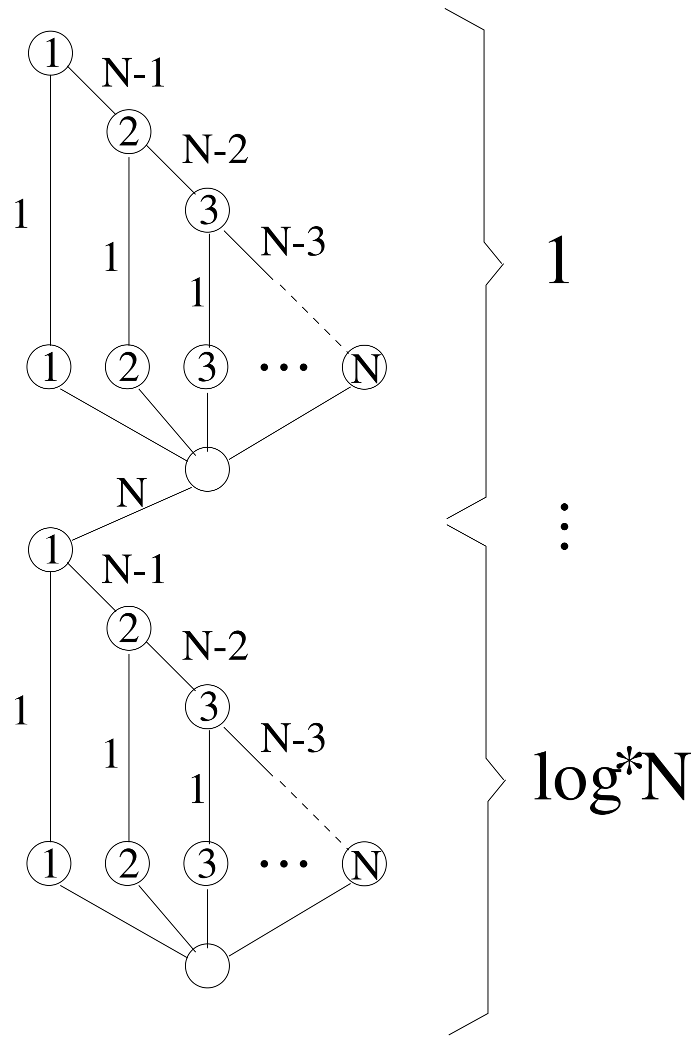

Figure 3: An example that flow-cached OSPF-R can be very bad

congestion factor is $\frac{N}{2}$. Therefore, we have established an $\Omega(N)$ lower-bound for the ratio between the performance of RS-R_1 and OSPF-R under the Congestion Factor criterion as well.

4.5 Low diameter (single hop) topologies

The above performance bounds have been established based on topologies that had as much as $\Omega(N)$ hops. Since many typical network topologies have a much lower diameter, it is of interest to evaluate the relative performance of OSPF-R in such cases. Hence, we consider a single-hop network, composed of a source, a destination, and some L parallel links interconnecting them, denoted by $1, 2, \ldots, L$.

OSPF-R vs RS-R_1 under the Max Flow criterion: the basic case

We begin by establishing a $\Omega(\log N)$ *lower-bound* on the ratio between the performance of RS-R_1 and OSPF-R under the Max Flow criterion and the basic case.

Let the link capacities be $c_l = \frac{C}{L-l+1} \cdot \frac{1}{\ln L}$, $1 \le l \le L$, where C is equal to the throughput of the demand matrix $\mathcal{D}$. Clearly, RS-R$_1$ achieves a throughput of C. Consider now OSPF-R. Since link capacities are nondecreasing in the link index, maximum throughput is achieved by choosing some subset of links with maximum indexes, i.e., $l^*, l^*+1, \ldots, L$, for some $1 \le l^* \le L$. Since we consider here the basic (noncached) case, the corresponding throughput is

$$(L - l^* + 1) \cdot c_{l^*} = (L - l^* + 1) \cdot \frac{C}{L - l^* + 1} \cdot \frac{1}{\ln L} = \frac{C}{\ln L},$$

i.e., $\frac{1}{\ln L}$ times the throughput of RS-R$_1$ (for *any* choice of l^*).

Next, we show that $\Theta(\log N)$ is also an *upper-bound*. Specifically, we show that if the sum of capacities on the L links is C, then OSPF-R can always achieve a throughput of at least $\frac{C}{\ln L}$, i.e., for *any* allocation of C over the links. We have seen above that this is true for the capacity allocation $\{c_l\}_{l=1}^L = \{\frac{C}{L-l+1} \cdot \frac{1}{\ln L}\}_{l=1}^L$. Consider then a different allocation $\{\hat{c}_l\}_{l=1}^L$, and, without loss of generality, let $\hat{c}_i \le \hat{c}_j$ if $i < j$. Denote $\delta_l = \hat{c}_l - c_l$. If $\delta_l \equiv 0$ for all l then we are done; otherwise, as $\sum_{l=1}^L \delta_l = 0$, there must be some j, $1 \le j \le L$, such that $\delta_j > 0$. The throughput obtained by OSPF-R under $\{\hat{c}_l\}_{l=1}^L$ is lower-bounded by the amount it achieves by routing over the specific subset of links $j, j+1, \ldots, L$; the latter, in turn, is equal to

$$(L - j + 1) \cdot \hat{c}_j = (L - j + 1) \cdot (c_j + \delta_j) > (L - j + 1) \cdot c_j = \frac{C}{\ln L},$$

hence establishing the required upper-bound.

5 Algorithms for Setting Routes in IP Networks

5.1 Why static weights are not helpful?

There is a considerable amount of research devoted to assigning weights to links in OSPF in order to avoid congested links. The rational behind this approach is that by assigning weights one can overcome a concentration of flows along the minimum hop routes in the network. For example, consider a network where N nodes are connected to node 1 via node 2, but also have disjoint routes to node 1 that are all 3 hop long. If all the links have the same cost, shortest path routing (SPR) will result in a concentration of N flows in the link between node 2 and node 1. Assigning a weight of, say, 2.5 to this link will divert the flows to the N disjoint routes and will achieve an improvement of $O(N)$ in the maximum load. However, we now show that, in some cases, weight assignment cannot alleviate the maximum flow problem.

Specifically, we show that, by assigning static weights to links in the network and using shortest path routing (SPR) with these link weights, one can get a maximum load which is $O(N)$ worse than the optimal solution. Consider the network of Figure 4 where N flows has to be routed from the sources $1, 2, \ldots, N$ to the destinations $1', 2', \ldots, n'$. As before, all the flows are equal, and all the capacities are the same. Using SPR, only one route will be used between nodes x and y, while, since all the flows are destined to different nodes, they could be evenly spread among the N possible routes between x and y. Note that this observation applies not only to IP-R but also to the more general RS-R$_1$ paradigm.

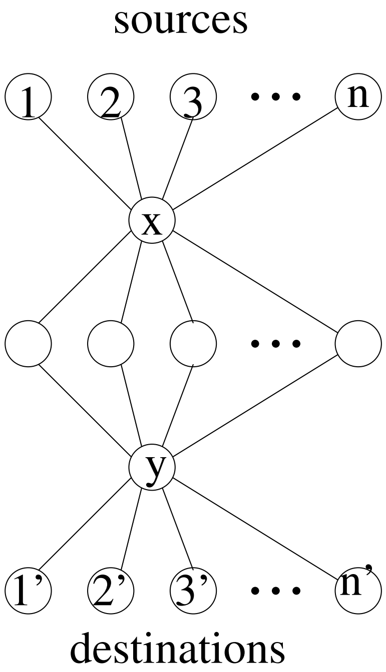

Figure 4: An example of the bad behavior of static link weight assignment

5.2 Algorithm description

Our aim is to improve the performance of centrally controlled IP networks. We showed above that the reason that SPR has such bad load ratio is that, once the weights of the links are determined, the routing is insensitive to the load already routed through a link. Thus, we propose a centralized algorithm that is given as input a network graph and a flow demand matrix. The demand matrix is built from long term gathered statistics about the flow through the network. Working off-line enables our algorithm to assigns costs to links dynamically while the routing is performed, thus achieving a significant improvement over other algorithms. The routing of each flow triggers a cost increase along the links used for the routing.

As link cost functions we chose the family $e^{-\alpha(c_e - flow_e)}$, which was found [AAP93] to exhibit good performance for related problems. (The function that was used in [FT00] is a piece-wise linear approximation of our function.) The parameter α determine how sensitive is the routing to the load on the link. For $\alpha = 0$, it is simply minimum hop routing which is load insensitive. For higher values of α, the routing sensitivity to the load increases. Clearly, if the routing is too sensitive to the load, it may prefer routes that are much longer than the shortest path, and the total flow in the network may increase. Thus, we seek a good trade-off between minimizing the maximum load in the network and minimizing the total flow.

Each flow is routed along the least cost route from the source to the destination, with the restriction that, if the new route hits another route to the same destination, the algorithm must continue along the previous route, as we assume IP forwarding. The calculation can be done using any SPR algorithm (with the above mentioned IP restriction); we chose to use the Bellman-Ford algorithm due to its efficiency and simplicity.

A potentially significant factor is the order by which flows are examined. We tested three heuristics:

rand - the flows are examined in random order.

sort - the flows between each source-destination pairs are cumulated, and then examined in decreasing order.

dest - the total flows to each destination are cumulated and then the flows to the destinations with more flows are examine first, with source weights used as the second sort key.

5.3 Performance evaluation

To test our algorithms, we generated two types of random networks, and two types of demand matrices. The network classes we generated were: **Inet**, i.e., preferential attachment networks that are now widely considered to represent the Internet structure [BA99, FFF99], and **flat**, i.e., Waxman networks [Wax88] which have been considered rather extensively in the literature and might better represent the internal structure of autonomous systems. For the flow demand matrix we selected the destination nodes uniformly among the network nodes. The source nodes were selected either uniformly, or according to a Zipf-like distribution, where the i-th most popular source is chosen with probability proportional to $1/i^{\alpha}$ (with $\alpha = 0.5$). The latter distribution was shown to model well the web traffic in the Internet [BCP+99].

The network links were assumed to have a unit capacity ($c_e = 1,\ \forall e \in E$), while the flows were assumed to have infinite bandwidth requirements, and thus each flow contributes a unit capacity to the demand matrix (Thus, $d_{i,j} \in \{0, 1, 2, 3, \ldots\}$. $d_{i,j}$ may be greater than one if more than one flow is selected between the same source-destination pair.). We tested the cost function $e^{-\alpha(c_e - flow_e)}$ with $\alpha = \beta/D,\ \beta = 0, 1, 20, 100, D$, where $D = \sum_{i,j} d_{i,j}$. Note that, when $\beta(= \alpha) = 0$, all the link costs are uniformly one and the algorithm performs minimum hop routing.

Figures 5 - 12 present the performance of the different heuristics for three loads: 200, 2000, and 20,000 flows, and for $\beta = 0, 20, 100$, and D. The results for $\beta = 1$ were omitted from the graphs since the algorithms performed almost identically for $\beta = 1$ and $\beta = 20$: the difference in the total network flow was close to zero, and the load on the most congested link was identical or slightly higher for $\beta = 1$. All the bars in the graphs represent an average of 25 executions that are the result of applying five random demand matrices on five random network topologies.

Figures 5 - 8 show the load on the most congested link for ten combinations of the three heuristics and β values. The most obvious result in these figures is that, even when a mild dependency on the link load is used ($\beta = 20$ and the same holds for $\beta = 1$, which is not shown), the load on the most congested link decreases significantly. For high demand (20,000 flows), the decrease is greater than 65% for the Inet networks, and 16.5% and 43% for flat networks. Even for very low demand (200 flows) the decrease in the maximum load is over 13%, and in many cases close to 50%. As we increase β, the gain increases accordingly.

Figures 9 - 12 show that the improvement in the reduction of the load comes at a cost that is negligible for all β values up to 100. Only when we set $\beta = D$ we see a significant increase in the traffic in the network. The difference between the heuristics for the order at which the flows

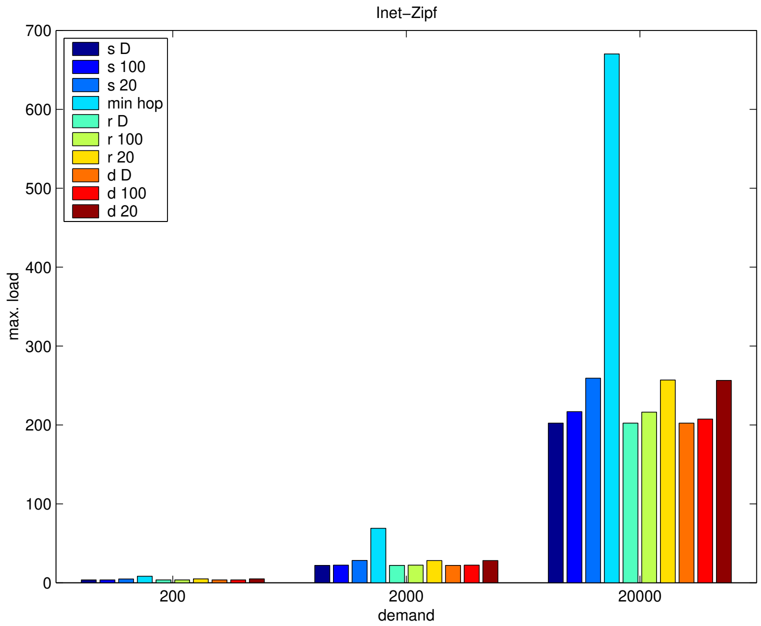

Figure 5: The load on the most congested link. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

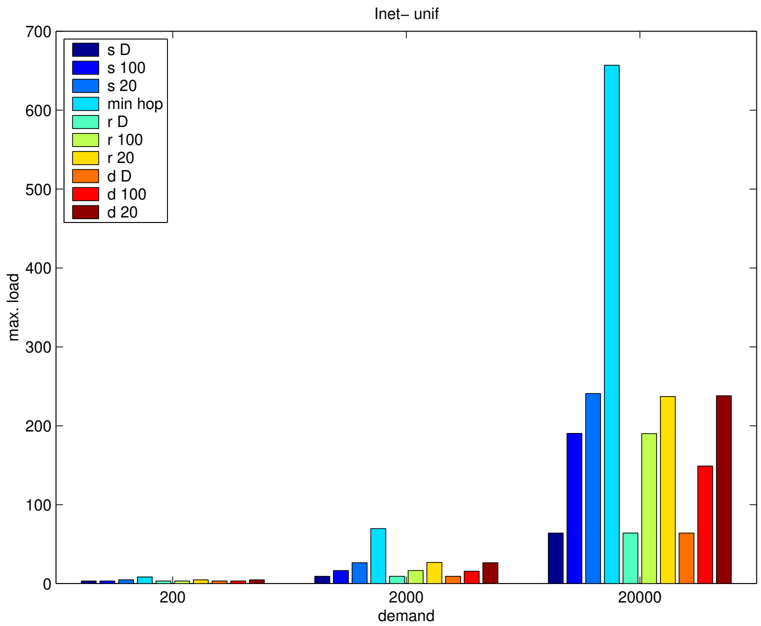

Figure 6: The load on the most congested link. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

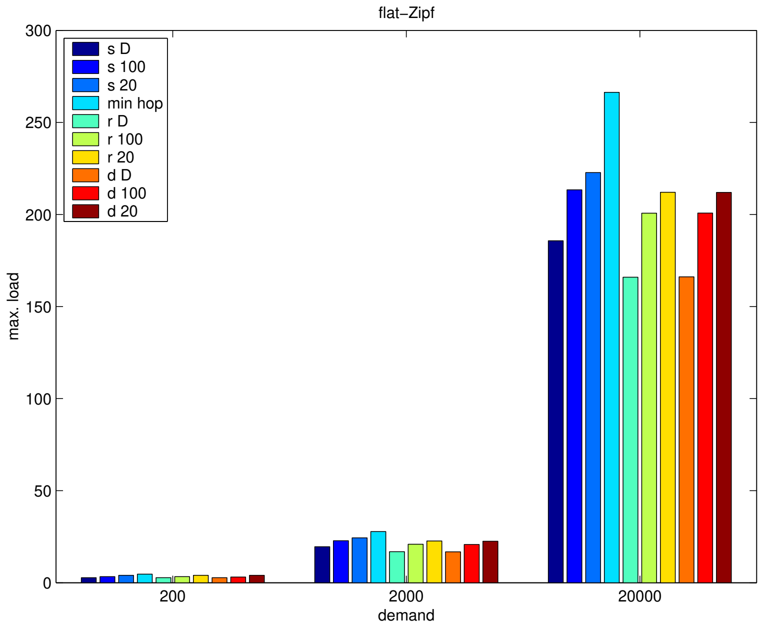

Figure 7: The load on the most congested link. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

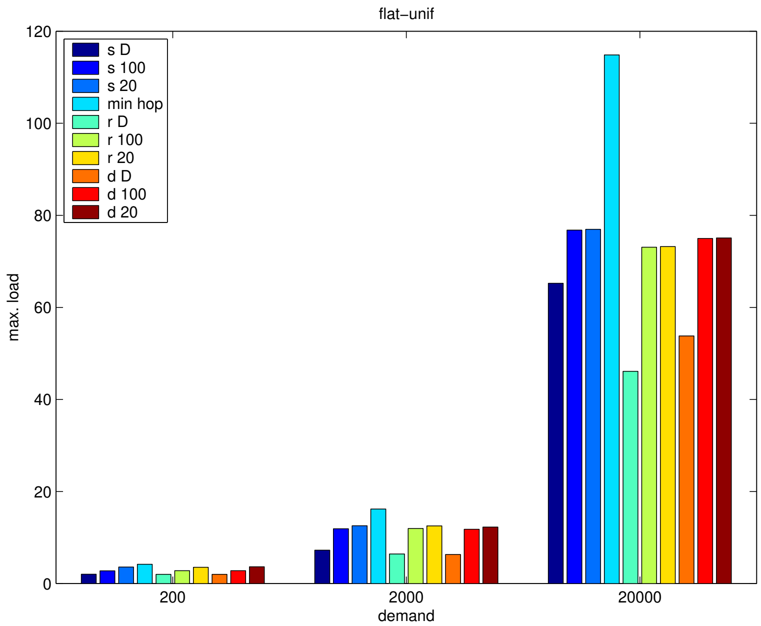

Figure 8: The load on the most congested link. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

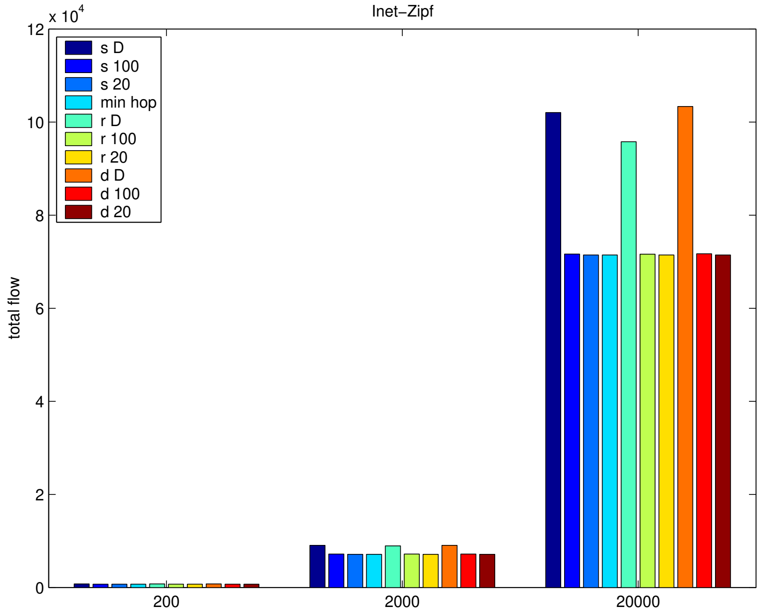

Figure 9: The total flow in the network. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

are examined by the algorithm are not significant; surprisingly, random order proved to be the best policy.

Thus, we can conclude that exponential dynamic link cost functions significantly improve the network performance. In addition, the performance of our algorithmic scheme is relatively insensitive to the scale factor in the exponent of the link cost function.

Figures 13 - 15 offer a closer look at the algorithm's performance. These figures present histograms, where bin i holds the number of links with load between $10(i-1)+1$ and $10i$; bin 0 holds the number of unused links and bin 31 holds all the links with load over 300.

Figure 13 compares the link load distribution for the Inet topologies and Zipf source distribution. Recall that, for this combination, our heuristics shows a significant improvement over minimum hop routing (see Figure 5). The left hand side graph is an unscaled plot of the bins, while the right hand side was scaled to present the differences in the bins that hold links with high load. Looking at the scaled graph, it is clear that, for all β values, our algorithm significantly reduces the number of links with high load. This is even more vivid for the last bin that holds the links with extreme load values. For $1 \leq \beta \leq 100$, the difference between the histograms is very small (while there is a difference in the maximum load value, see Figure 5), but for $\beta = D$ it is clear that the reduction in loaded links is bigger for loads over 70 flows per link. However, this gain is offseted by a very large increase in the number of links with loads 21-60, as can be seen in the left hand side histogram. Smaller β values shift the mode of the load distribution to the left.

Figure 14 compares the link load distribution for the Inet topologies and Zipf source distribution. For this combination, our heuristics exhibit the smallest improvement over minimum hop routing, around 16.5% (see Figure 7). Here the gain is not that vivid, but still noticeable for $1 \leq \beta \leq 100$.

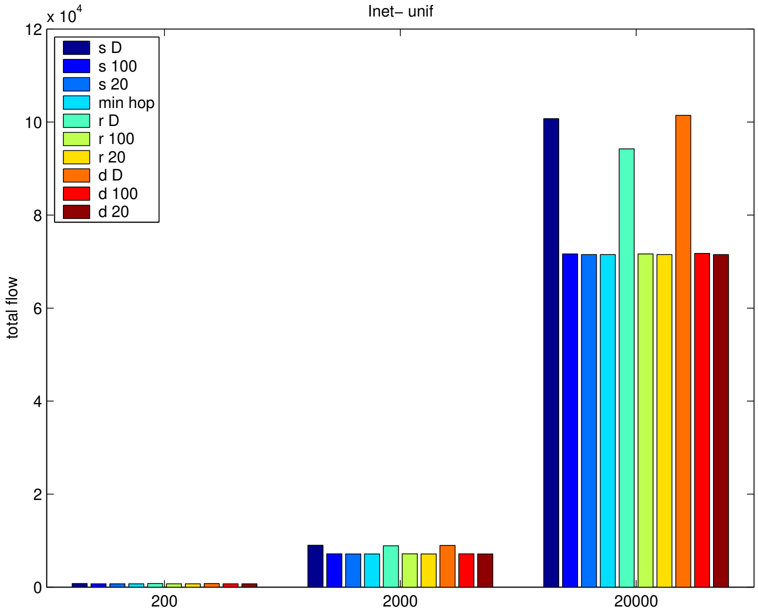

Figure 10: The total flow in the network. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

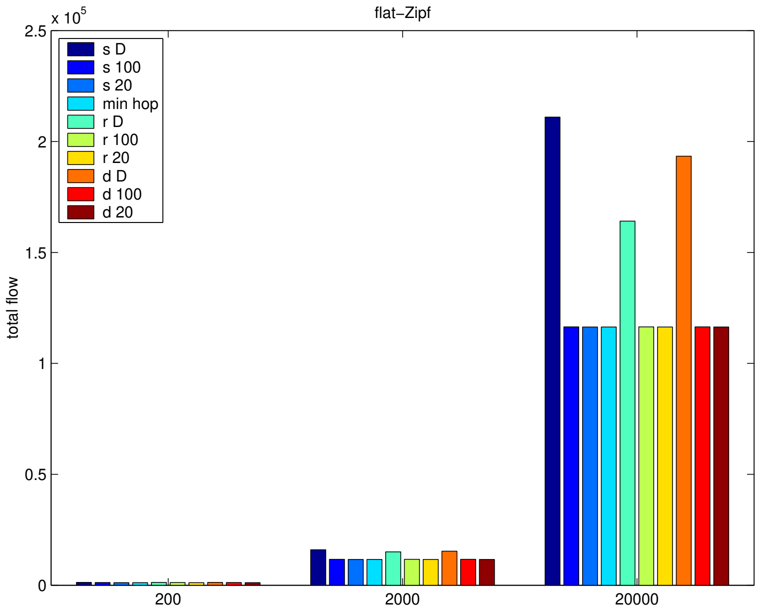

Figure 11: The total flow in the network. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

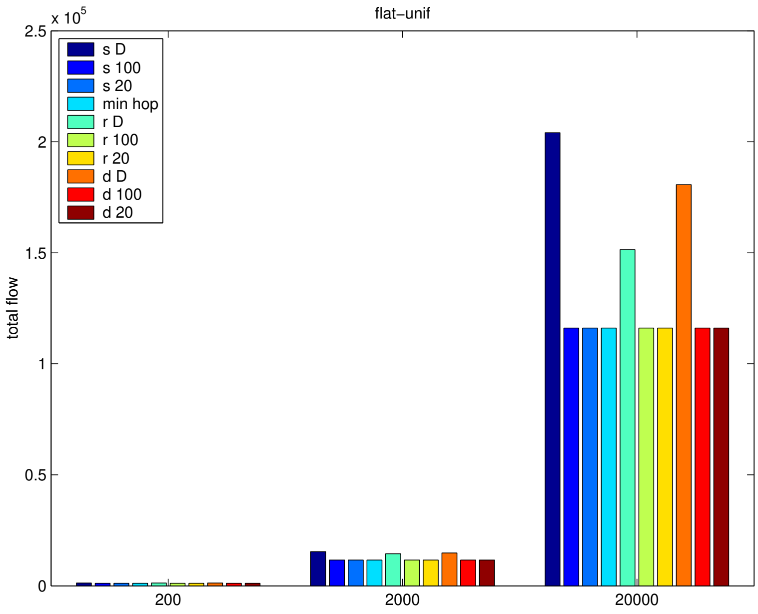

Figure 12: The total flow in the network. The first letter in the legend stands for the heuristic used (r for rand, d for dest, and s for sort) the second term is the value of β.

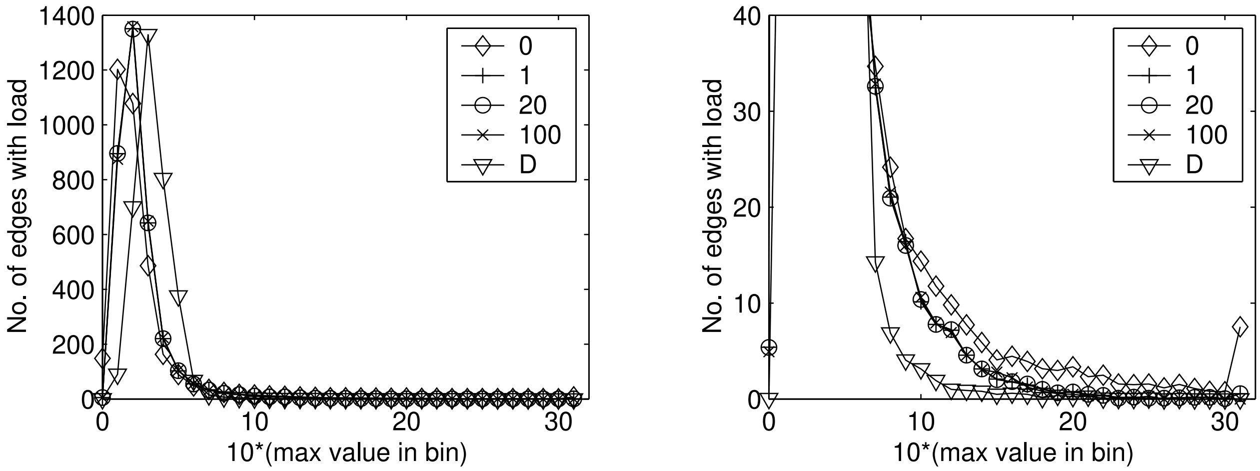

Figure 13: Histogram of the load on the links. The bucket width is 10. The mark at tick i depicts the number of links with load between $10(i-1)+1$ and $10i$. The histogram is for Inet networks with Zipf source distributions, demand of 20,000 flows, and the sort heuristic.

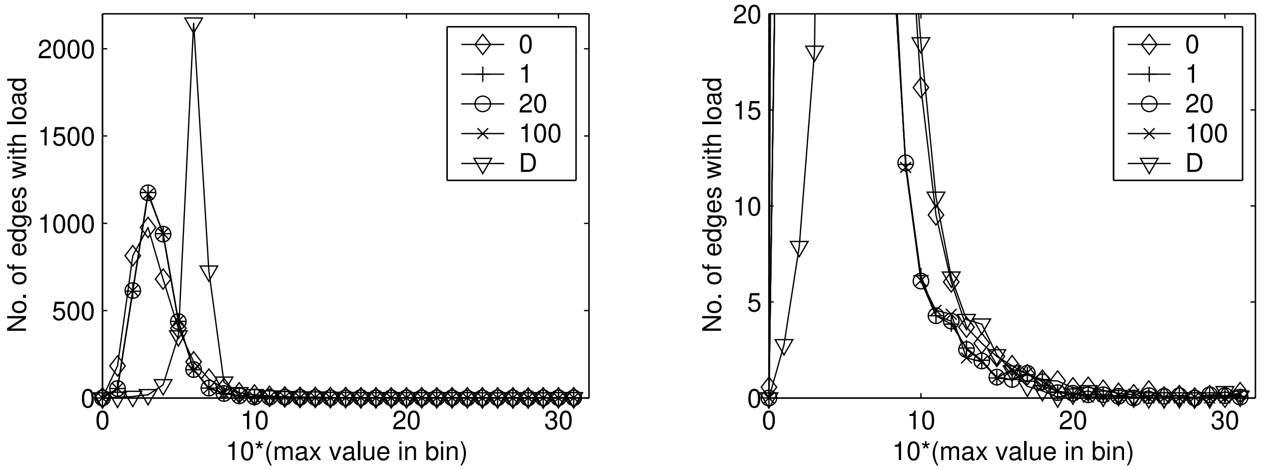

Figure 14: Histogram of the load on the links. The bucket width is 10. The mark at tick i depicts the number of links with load between $10(i-1)+1$ and $10i$. The histogram is for flat networks with Zipf source distributions, demand of 20,000 flows, and the sort heuristic.

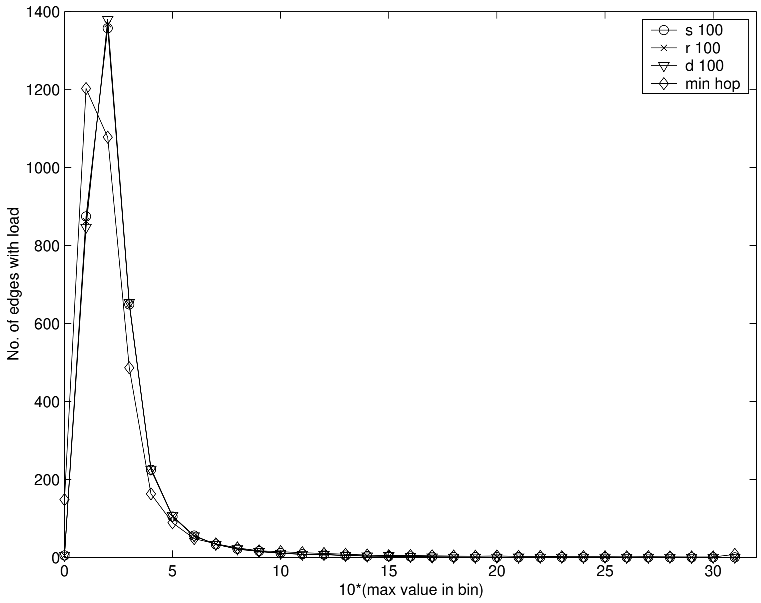

Figure 15: Histogram of the load on the links. The mark at tick i depicts the number of links with load between $10(i-1)+1$ and $10i$. The histogram is for Inet networks with Zipf source distributions and demand of 20,000 flows. It compares the three heuristics for $\beta = 100$ and minimum hop routing.

The increase in traffic is, however, much vivid from the left hand side graph, where the mode is shifted up and to the right. Figure 15 shows that the small difference between the heuristics appears also in the more detailed histogram view.

6 Concluding Remarks

Distributed load sensitive routing was abandoned in the Internet due to the instability it introduced [KZ89, BG92]. Shaikh et al. [SRS99] suggested to use load sensitive routing for OSPF intranets; to avoid the stability problem, they advocated to use load sensitive routing only for long lived flows. The routing rule they used is to select the shortest path with sufficient capacity. Fortz and Thorup [FT00] studied the optimal allocation of link weights for OSPF, but their study is limited to a *specific* cost function. Here, we suggested to use a centralized non-interactive approach, which is load sensitive in the sense that it takes into account the forcasted load in the network.

Although we showed that our routing algorithm is very effective in reducing the load on the network link, it is only a first step in this direction and there is much room for improvement.

In particular, we believe that the basic algorithm can be augmented with step-wise improvements via rerouting. For example, once routing is done for all links, we can select a flow that uses the most loaded link, remove it, and reroute it. This process can continue until no further improvement is achieved.

References

[AAP93] Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput-competitive on-line routing. In *34th Annual IEEE Symposium on Foundations of Computer Science*, pages 32 – 40, October 1993.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *SCIENCE*, 286:509 – 512, 15 October 1999.

[BCP+99] Lee Breaslau, Pei Cao, Li Pan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99*, pages 126 – 134, March 1999.

[BG92] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

[DGG99] Ye. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19:17–41, 1999.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM 1999*, August 1999.

[FT00] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. In *IEEE INFOCOM 2000*, pages 519 – 528, Tel-Aviv, Israel, March 2000.

[GJ79] Michael R. Garey and David S. Johnson. *Computer & Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman, November 1979.

[Hed88] C. Hedrick. Routing Information Protocol, June 1988. Internet RFC 1058.

[IPN] The IP Network Configurator. `www.lucent.com/OS/ipnc.html`.

[KZ89] Atul Khanna and John Zinky. The revised ARPAnet routing metric. In *ACM SIGCOMM'89*, pages 45–56, Austin, TX, September 1989.

[Mal98] G. Malkin. RIP Version 2, November 1998. Internet RFC 2453.

[Moy95] John Moy. Link-state routing. In Martha E. Steenstrup, editor, *Routing in Communications Networks*, pages 135 – 157. Prentice Hall, 1995.

[Moy98] John Moy. OSPF Version 2, April 1998. Internet RFC 2328.

[MS95] Gary Scott Malkin and Martha E. Steenstrup. Distance-vector routing. In Martha E. Steenstrup, editor, *Routing in Communications Networks*, pages 83 – 98. Prentice Hall, 1995.

[MSZ97] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. In *ACM SIGCOMM'96*, pages 206–217, Stanford, CA, August 1997.

[PNN96] Private network-network interface specification version 1.0 (PNNI). Technical report, The ATM Forum technical committee, March 1996. af-pnni-0055.000.

[Pos81] J. Postel. Internet Protocol, September 1981. Internet RFC 791.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture, January 2001. Internet RFC 3031.

[SRS99] A. Shaikh, J. Rexford, and K. Shin. Load-sensitive routing of long-lived ip flows. In *ACM SIGCOMM'99*, pages 215–226, Cambridge, MA, September 1999.

[Wax88] Bernard M. Waxman. Routing of multipoint connections. *Journal on Selected Areas in Communications*, 6:1617 – 1622, 1988.