Routing with Load Balancing - Increasing the Guaranteed Node Traffics

Marija Antić, *Graduate Student Member, IEEE,* and Aleksandra Smiljanić, *Member, IEEE*

***Abstract*—In this paper we introduce the novel routing scheme based on load balancing and shortest-path routing. First, we present the linear program for routing optimization. The non-blocking network is considered, which only limits the traffic loads of the network nodes. Guaranteed node traffic loads pass through the network regardless of the actual traffic destinations. The derived optimization includes node priority weights that allow the network planner to assign higher or lower traffic values to the network nodes. Then we analyze the performance of the proposed strategy for some realistic network topologies, and show that the proposed scheme achieves higher guaranteed node traffic loads than the regular shortest-path routing.**

***Index Terms*—Routing, load balancing, linear programming**

I. Introduction

THE peer-to-peer traffic is becoming dominant on the Internet today, making it a highly dynamic environment. The actual traffic distribution becomes very hard to predict and this requires certain changes in the routing design. Usually, the routing scheme in a network is oblivious, i.e. the path for the traffic between the origin-destination (OD) node pair is predetermined, and the routing decisions are made based only on the traffic destination, while the activity of the nodes in the network and the actual link loads are not taken into consideration. Commonly, the traffic pattern is described by the traffic matrix $\mathbf{TM} = [d_{ij}]_{N\times N}$, whose elements d_{ij} represent the traffic load between the node pairs (i, j). Even the best existing techniques for the traffic matrix prediction involve serious error margins. The increasing peer-to-peer traffic makes this prediction even more difficult.

The maximum number of users that can be serviced by a network node is proportional to the guaranteed (or permissible) node traffic, i.e. the traffic that can always be routed through the network. We will observe the guaranteed node traffic as the performance measure of a particular routing scheme on the given network, and define the optimal routing as the one that maximizes the guaranteed node traffic. The problem of finding the optimal oblivious routing was addressed by many authors. In [1], a non-polynomial solution for general networks, based on the graph decomposition was proposed. Later it was followed by the polynomial-time algorithm in [2], [3]. In [4], [5] the authors use linear programming to optimize the routing strategy. In [6], [7] Kodialam et al. introduce the two-phase routing scheme, based on load balancing. Load balancing allows them to calculate the elements of the traffic matrix based on the incoming/outgoing node traffic loads, which are easier to predict. The linear program introduced in [7] considers the general case of routing coefficients' optimization, and has $O(N^2M)$ variables and $O(N^2M)$ constraints (where N and M are the number of nodes and links in the network, respectively). Such high complexity of this linear program makes it impossible to use in practice, where recalculations are triggered every time the network topology changes, and have to finish within minutes. The experiment that we ran, and that will be described in this paper, showed that the optimization in [7] takes as long as several days even for the smallest of the analyzed networks.

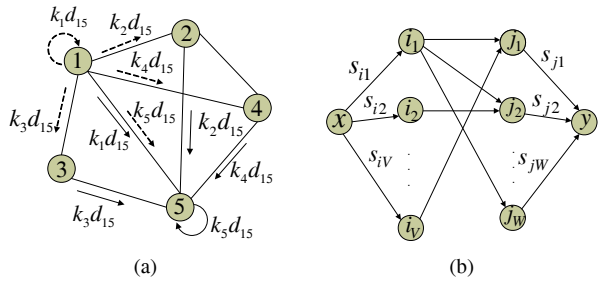
The routing scheme that we analyze in this paper is similar to the one proposed by Kodialam et al. We use load balancing and route every flow in two stages. But instead of solving the general problem of routing coefficients' optimization, we decide to use the shortest-path routing in both of the stages. This allows us to significantly simplify the linear program - we only optimize the node weights that determine the traffic split ratio for load balancing. Our model, therefore, has only $O(N)$ variables and $O(M)$ constraints, compared to $O(N^2M)$ variables and constraints in [4], [5], [7]. The proposed routing was already introduced in [9], but only the case when all the nodes in the network generate and receive equal traffic loads was considered. Now, we introduce the node weights in the linear program in a way that allows us to guarantee different traffic loads to different nodes, which are proportional to the preassigned weights. Namely, in a real network, the traffic loads are not necessarily equal for all the nodes. We consider two different weight models, with unequal node demands. It turns out that the proposed routing scheme performs very well in the analyzed situations, and allows significantly higher guaranteed node traffic loads, compared to the case of the shortest-path routing commonly used in the networks today.

II. Load Balanced Routing (LBR)

It has been shown that the load balancing increases the capacity of the non-blocking regular networks [8]. It is used to distribute the traffic evenly among the links in the network and avoid bottlenecks. Also, it allows to simplify the linear program for the optimal routing design, and express it only in terms of the total incoming/outgoing node traffic loads.

The traffic between a node pair (i, j) is routed in two steps. First, portions of the flow from i are routed to intermediate nodes $m \in V$ (V is the set of network nodes). In the next step, every intermediate node forwards the traffic to its final destination j. The traffic from i to m, i.e. from m to j is routed along the shortest paths. The portion of the flow that is

M. Antić is with the Faculty of Electrical Engineering, Belgrade, Serbia.
A. Smiljanić is with the Faculty of Electrical Engineering, Belgrade, Serbia and the Polytechnic Institute of New York University.



(a) (b)

Fig. 1: (a) Routing scheme illustration. (b) The assigned graph.

balanced across node m equals k_m, and does not depend on i and j. Of course, $\sum_{m\in V} k_m = 1$. Fig. 1a illustrates the case of routing the traffic between the nodes 1 and 5. The first step flows are represented by the dashed arrows, and the second step flows by the solid ones.

The node traffic maximization and the congestion minimization problems are equivalent. Congestion Q represents the maximum link utilization in the network, where the link utilization is the ratio of the link load $L(l)$ and the link capacity, $C(l)$. It is related to the maximum guaranteed traffic, since all the initial flows in the network can be increased up to $1/Q$ times, without causing the link overload.

Assume that the network is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. Let the number of nodes be N and the number of links M. Let i be a source node. The outgoing traffic generated by i equals $s_i = \sum_{j\in V} d_{ij}$, where d_{ij} is the intensity of a flow from i to j. Similarly, for j being the destination node and r_j its total incoming traffic, $r_j = \sum_{i\in V} d_{ij}$. We have already defined the *traffic matrix* as $\mathbf{TM} = [d_{ij}]_{N\times N}$. We will refer to the vectors $\mathbf{S} = [s_1, s_2, \ldots, s_N]$ and $\mathbf{R} = [r_1, r_2, \ldots, r_N]$ as *out-traffic vector* and *in-traffic vector*.

As stated before, the load balancing allows us to formulate the problem of finding the optimal routing strategy in terms of the total node traffic. Namely, the traffic between any two nodes i and m consists of two components: the traffic $b_{im}^{(1)}$, generated by i and balanced across m, and the traffic $b_{im}^{(2)}$, directed to m and passing through i as the intermediate node. It is easy to see that it holds

$$b_{im}^{(1)} = \sum_{j\in V} k_m d_{ij} = k_m s_i \tag{1}$$

$$b_{im}^{(2)} = \sum_{p\in V} k_i d_{ip} = k_i r_m. \tag{2}$$

The load of link $l \in E$ can be expressed as

$$L(l) = \sum\nolimits_{(i,m)} F_{im}^{(l)}(k_m s_i + k_i r_m), \tag{3}$$

where $F_{im}^{(l)} = 1$ if the link l is on the shortest path between i and m, and $F_{im}^{(l)} = 0$ otherwise. Obviously, the link load does not depend on the traffic pattern between nodes, but only on the total incoming/outgoing node traffic.

We will assume that incoming and outgoing traffic loads at the nodes are equal, i.e. $r_i = s_i$ for every i. This assumption is realistic in the backbone, regional and even local networks, since these networks utilize the equipment with symmetric bidirectional links, unlike the access networks which often use the equipment with asymmetric links and which are not considered here. As a result, we can use the following linear program to minimize the congestion:

$$\begin{aligned} &\min Q \\ &\text{(C1)}\ \textstyle\sum_{i=1}^{N} k_i = 1 \\ &\text{(C2)}\ \forall l \in E:\ \frac{\sum_{(i,m)} F_{im}^{(l)} (k_i s_m + k_m s_i)}{C(l)} \le Q \end{aligned} \tag{4}$$

The guaranteed incoming and outgoing traffic will be equal to the in-traffic and out-traffic vector divided by the calculated minimal link congestion. By setting the in- and out-traffic vector elements to be equal to the desired node weights, $w_i, i \in V$, we can control the proportion of the guaranteed node loads. We will examine two different cases for assigning the weights.

1) Case I: First, we assume that the node traffic is proportional to the total link capacity entering/leaving the node. Let the total link capacity entering and leaving the node i be $C_n(i)$. The values of the in-vector and out-vector elements are:

$$s_i = r_i = w_i = \frac{C_n(i)}{\min_{j\in V} C_n(j)}. \tag{5}$$

2) Case II: We suppose that the in-vector and out-vector traffic values are proportional to the number of inhabitants serviced by the network node. When there are multiple routers (nodes) at the same location, we assume that the total load is equally distributed among them. Let $p(z)$ denote the population at the location z, and $n(z)$ the number of routers. The population serviced by a router i at this location equals $w_i' = p(z)/n(z)$. We assume that $r_i = s_i$ for every i and that the out-traffic vector elements have the following values:

$$s_i = w_i = \begin{cases} 1, w_i' \le 10^5 \\ \left\lfloor w_i'/10^5 \right\rfloor, u \le \frac{w_i'}{10^5} \le u + 0.5, u \in N \\ \left\lfloor w_i'/10^5 \right\rfloor + 1, u + 0,5 \le \frac{w_i'}{10^5} \le u + 1, u \in N \end{cases} \tag{6}$$

Here, $\left\lfloor w_i'/10^5 \right\rfloor$ is the floor function, and u an integer.

The guaranteed traffic load for the node i in the network is proportional to the value of the out-vector element s_i, and equals s_i/Q. The values of the in-vector and out-vector elements are chosen so that $\min_i s_i = \min_i r_i = 1$. So, the minimum guaranteed node traffic load equals:

$$s_{gtd}^{lbr} = 1/Q. \tag{7}$$

III. Shortest Path Routing (SPR)

We determine the guaranteed node traffic for shortest-path routing (SPR), in order to compare it with the guaranteed node traffic for the proposed load balanced routing (LBR). In the case of the SPR, the link loads depend on the traffic-matrix elements , and not only on the node traffic loads. The worst case traffic-patterns should be found for all links, and they determine the guaranteed node traffic loads

Let us denote the set of all node pairs that communicate across the link $l \in E$ as $P(l) = \left\{(i,j) | F_{ij}^{(l)} = 1\right\}$. Define

TABLE I: Results for the six Rocketfuel network topologies

NETWORK	N (nodes)	M (links)	Model Size var	Model Size con	Case I G	Case I t[sec]	Case II G	Case II t[sec]
AS 3967 (Exodus)	79	294	80	282	6.02	2.77	5.74	2.88
AS 1755 (Ebone)	87	322	88	316	2.02	3.38	3.36	3.56
AS 1221 (Telstra)	104	302	105	304	2.24	3.53	2.34	4.12
AS 6461 (Abovenet)	138	744	139	720	4.93	9.36	3.04	9.22
AS 3257 (Tiscali)	161	656	162	629	5.95	15.92	7.99	13.38
AS 1239 (Sprintlink)	315	1944	316	1923	7.69	60.36	5.57	61.42

the set of sources sending their traffic across l by $I(l) = \{i|\exists j, (i,j) \in P(l)\}$, and the set of destinations receiving the traffic across l by $J(l) = \{j|\exists i, (i,j) \in P(l)\}$.

Let us calculate the traffic loads that can be guaranteed when SPR is used in a given network. The maximum matching algorithm cannot be used in this case as in [9] because the nodes generate different traffic loads. We form a bipartite graph with $I(l)$ and $J(l)$ as the sets of nodes - Fig. 1b. Let there be a directed edge of infinite capacity in the bipartite graph between every pair of nodes $(i,j) \in P(l)$ (note the difference between these edges and the links in the network, they are not related). Let us now add the additional source and sink nodes to this graph. Let the source node be denoted by x and the sink node by y. Let there be a directed edge from x to every $i \in I(l)$, with the capacity s_i. Also, let there be a directed edge from every $j \in J(l)$, with the capacity s_j. Now we determine the value of the maximum flow from x to y in this graph, $f_{max}(l)$, i.e. the maximum traffic load that the node x can generate and pass to y through the given network [13]. The worst-case link utilization is $U(l) = f_{max}(l)/C(l)$. Since $\min_i s_i = 1$, this implies the node traffic limit on link l to be $1/U(l)$. The minimum guaranteed node traffic in the whole network equals

$$s_{gtd}^{spr} = \min_{l \in E} 1/U(l). \tag{8}$$

IV. Results

We compared the performance of LBR with the performance of SPR on the six backbone network topologies published in the *Rocketfuel* project [10], for two different node weight models, described in the previous section. We calculated the gain of the guaranteed node traffic loads when balancing is used:

$$G = s_{gtd}^{lbr}/s_{gtd}^{spr}. \tag{9}$$

Since the original data did not include the link capacities, but only the link weights, we assumed that the capacity of the link is inversely proportional to the link weight. For the analysis in Case II, we also needed the information on the number of inhabitants in the cities where the routers are located [11]. We used the linear program (4) to optimize the node coefficients and obtain the maximum guaranteed node traffic for the LBR. Then we calculated the guaranteed node traffic in the case of SPR, as described in Section III, and calculated the gain (9).

The gains and the times needed to optimize the node coefficients are given in Table I. The gain depends on the network topology - for the more meshed topologies, the gain results are higher. For the Case I the gain ranges from 2.02 to 7.69, and for the Case II from 2.34 to 7.99. The optimization times listed in the table are for the AMD Athlon 64 3500+ processor (2.2 GHz clock speed, 960 MB of RAM), using the *LP_Solve* optimization software [12]. For the Exodus network (AS 3967) we also tested the linear program proposed in [7] - the optimization lasted more than five days, compared to less than three seconds in our case.

V. Conclusion

We showed that the load balancing significantly increases the guaranteed node traffic load in the real case network topologies. It also allows to express the routing optimization problem only in terms of the total node traffic loads, which are easier to predict than the traffic loads between node pairs. The complexity of the linear program used to optimize the routing coefficients is acceptable, and the proposed routing strategy could easily be brought to life. It represents only a minor modification of the shortest-path routing, which is most widely used. When a packet enters the network, the intermediate node for it has to be decided. The packet is then routed along the shortest paths from the origin to the intermediate node, and from the intermediate node to the final destination. By assigning different weights to the nodes, the network planner can split the available network capacity to the nodes proportionally to the bandwidth demands at these nodes.

References

[1] H. Räcke, "Minimizing Congestion in General Networks," *FOCS 43*, 2002.

[2] C. Harrelson, K.Hildrum, and S. Rao, "A Polynomial-Time Tree Decomposition to Minimize Congestion," in *Proc. of SPAA'03*, 2003.

[3] M. Bienkowski, M. Korzeniowski, and H. Räcke, "A Practical Algorithm for Constructing Oblivious Routing Schemes", *Proc. of SPAA'03*, 2003.

[4] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs," *Proc. of SIGCOMM '03*, 2003.

[5] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke, "Optimal Oblivious Routing in Polynomial Time," *Proc. of the 35th ACM Symposium on the Theory of Computing*, 2003.

[6] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-Oblivious Routing for Guaranteed Bandwidth Performance," *Communications Magazine*, 45(4):46-51, Apr. 2007.

[7] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Maximum Throughput Routing of Traffic in the Hose Model", *Proc. of INFOCOM 2006*, 2006.

[8] A. Smiljanić, "Rate and Delay Guarantees Provided by Clos Packet Switches with Load Balancing", *Transactions on Networking*, Feb. 2008.

[9] M. Antić, A. Smiljanić, "Oblivious Routing Scheme Using Load Balancing Over Shortest Paths", in *Proc. of ICC 2008*, May 2008.

[10] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring Link Weights Using End-to-End Measurements," *Proc. of the IMW '02*, 2002.

[11] *Wikipedia, the Free Encyclopedia*, [Online] http://en.wikipedia.org

[12] *LpSolve, Reference Guide* [Online]http://lpsolve.sourceforge.net

[13] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[14] J. Matoušek, B. Gärtner, *Understanding and Using Linear Programming*, Springer Verlag, 2007.