Two Phase Load Balanced Routing using OSPF

Marija Antić, Nataša Maksić, Petar Knežević, and Aleksandra Smiljanić

Abstract*—The Internet traffic is growing, and its nature changes because of new applications. Multimedia applications require bandwidth reservations that were not needed initially when the file transfers dominated the Internet. P2P applications are making traffic patterns impossible to predict, and the traffic loads generated at nodes need to be routed regardless of the traffic pattern. When the guaranteed node traffic loads are known, bandwidth reservations can be made simple as will be explained in the paper. The shortest path routing (SPR) protocols used on the Internet today do not maximize the guaranteed node traffic loads, and do not provide scalable and fast bandwidth reservations. Load balancing can improve the network throughput for arbitrary traffic pattern. In this paper we analyze and implement a routing protocol that is based on load balancing and a commonly used shortest path routing protocol, and is, consequently, termed as LB-SPR. LB-SPR is optimized for an arbitrary traffic pattern, i.e. it does not assume a particular traffic matrix. Optimization assumes only the weights assigned to the network nodes according to their estimated demands. It will be shown that the optimized routing achieves the throughputs which are significantly higher than those provided by the currently used SPR protocols, such as OSPF or RIP. Importantly, LB-SPR calculates the guaranteed traffic loads and so allows fast autonomic bandwidth reservations which are the key for the successful support of triple-play applications, including video and audio applications that require high QoS. An actual modification of the TCP/IP stack that includes LB-SPR is also described. Using the signaling mechanisms of the OSPF protocol, the information needed to perform the routing optimization is automatically distributed among the network nodes whenever the network topology changes. The LB-SPR implementation is validated on a sample network using a popular virtualization tool - Xen.

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

I. Introduction

THE INTERNET traffic has experienced some major changes lately, which require modifications in the network planning and routing protocols. Heavy traffic loads are generated by the multimedia applications, and the actual traffic distribution in the network becomes very hard to predict, due to the developing peer-to-peer services. On the other hand, the traditional approach to traffic grooming and routing optimization in the optical networks assumes that the traffic demands between pairs of nodes are known, which is often not the case. New routing protocols should be able to optimally utilize the network without knowing the actual traffic distribution.

Manuscript received 1 December 2008; revised 31 May 2009. This work was funded by the Ministry of Science and Technological Development, Republic of Serbia.

M. Antić and N. Maksić are with the School of Electrical Engineering, Belgrade University, Serbia,

P. Knežević is with IRITEL, Serbia,

A. Smiljanić is with the School of Electrical Engineering, Belgrade University and the Polytechnic Institute of NYU.

Digital Object Identifier 10.1109/JSAC.2010.100106.

It is widely accepted that the next-generation networks should become more autonomic in the process of the network configuration, topology change detection and adaptation to the traffic load changes. Some of these features are incorporated into today's IP networks: they have the ability to detect the topology changes and change the routing accordingly, the TCP congestion control mechanism adapts the transmission speed to the traffic load changes, etc. However, the autonomic handling of multimedia applications is not resolved yet. These applications require high quality of service: bandwidth reservations and delay guarantees. Centralized bandwidth reservations can obviously become a bottleneck in large-scale networks, as well as the reservations which require each router to know about the available link capacities in the whole network. So, a new mechanism for fast bandwidth reservations is needed.

Because of the traffic unpredictability, the customers attached to the network nodes should be served regardless of the traffic pattern between them. In other words, the guaranteed node traffic loads should be sufficient to support all the users attached to these nodes. When the guaranteed node traffic loads are determined, the bandwidth reservations through the network become simple. Each session learns from its router (node) if it can be passed through the network, since the router knows its guaranteed traffic load and the already reserved capacity. If the session can be passed, its request for the bandwidth reservation is passed to the destination router, which checks if there is sufficient capacity on its links toward customers since it knows its guaranteed traffic load and the already reserved capacity. In this way, bandwidth reservations are distributed and are consequently agile. For each session, only two edge routers check their available capacities. And, each router handles bandwidth reservation only for the flows that are either entering or leaving the network through that router. Fast automated bandwidth reservations are very important for growing multimedia applications that demand high QoS, i.e. bandwidth and delay guarantees. If all the flows of equal priority negotiate certain policing interval, the delay guarantees can be achieved when the bandwidth is reserved.

It has been shown that the guaranteed node traffic loads can be significantly increased in the regular networks when load balancing is used [1]–[3]. Then, Kodialam et al. proposed to use two-phase routing in an arbitrary network, and showed significant improvements of the network throughput [4], [5]. However, this two-phase routing protocol is rather complex since it optimizes the balancing coeffticents using the linear program with $O(MN^2)$ variables and constraints. For this reason, we are proposing a novel two-phase routing, LB-SPR, where each phase uses the standard shortest path routing (SPR) protocol. In this way, we significantly decrease the complexity of the optimization problem to only $O(N)$ variables and $O(M)$ constraints, and make it practical. Also, LB-SPR can be implemented as a modification of the existing SPR protocols such as OSPF and RIP. Balancing mechanism itself does not increase computational complexity, it can be implemented by the loose source routing and a slight modification of the IP lookup algorithm.

In LB-SPR, every packet is routed in two phases, with SPR as the underlying routing protocol in both of the phases. When a packet arrives to the source router, its intermediate router is determined. The packet is sent to the intermediate router using the standard SPR protocol, and from the intermediate router to the destination router again using the standard SPR protocol. The load is balanced across the intermediate routers, meaning that the specified portions of each flow are transmitted through the intermediate routers. These portions are referred to as the balancing coefficients. A balancing coefficient depends only on the associated balancing router. Balancing coefficients are optimized to maximize the network throughput while ensuring that nodes can generate and receive loads which are proportional to the allocated weights. The node weights are chosen to reflect the expected demands at the nodes, as in [6]. For example, London is bound to generate more traffic than Luxembourg considering its size, and should be allocated the larger portion of the network capacity. This is achieved by allocating to London the higher weight than to Luxembourg.

We implemented the LB-SPR protocol using the programming language C++, and integrated the implemented protocol into the TCP/IP stack [7]. The LB-SPR protocol uses the signaling of the OSPF protocol. Through this signaling, each router in the network is learning the network topology, and the capacity of the nodes' external (customer) links. The external link capacities are taken to be the node weights. Consequently, LB-SPR maintains autonomic fault recovery mechanism developed within OSPF. Namely, whenever there is a network topology change, the routing is adjusted accordingly. LB-SPR is advantageous during the transient periods after failures because only the portions of the flows are affected, which can be tolerated by most Internet applications. When the network topology is learned, the linear program is run, and the optimal balancing coefficients are determined at each router. Now, the packets are easily routed based on the calculated balancing coefficients, and using the SPR protocol.

The paper is organized as follows. The second section represents the study of the related work. In the third section, the LB-SPR protocol is described. In this section, the linear program for the optimization of the balancing coefficients is also presented and simplified. Using this linear program, we analyze the performance of the LB-SPR protocol, i.e. the node traffic loads that it guarantees and its speed. In the fourth section, the performance of the standard SPR protocol is analyzed. In the fifth section, the performance of the LB-SPR and the SPR protocols (e.g. OSPF or RIP) are compared. In the sixth section, the implementation of the LB-SPR protocol is described. In the seventh section, the correct functioning of the LB-SPR protocol is confirmed through its simulation in the network of the virtual routers on several interconnected computers. The eighth section concludes the paper.

II. Related Work

Current routing protocols are typically oblivious, i.e. the paths for the traffic between source/destination node pairs are determined in advance. In this paper we will also consider oblivious routing protocols, because making the routing adaptive to frequent traffic load changes can have some serious side-effects: it can affect the quality of service and even the network connectivity [8]. In the currently used routing protocols, the routing decisions are made only based on the final packet destination. For certain traffic patterns, the current routing protocols can overload some links, while leaving the other links unused at the same time. This might limit the guaranteed node traffic loads, i.e. the traffic loads generated and received at the nodes that can be routed for arbitrary traffic distribution between pairs of nodes. Consequently, the currently used protocols might limit the number of users that can be serviced by the network more than necessary. To optimize the network performance it is, therefore, necessary to choose the routing protocol in such a way that the guaranteed node traffic loads are maximized. These guaranteed node traffic loads should not depend on the traffic pattern.

The optimization of the oblivious routing was addressed by many authors [9]–[14]. Unfortunately, all of these solutions rely on the previous knowledge or the prediction of the traffic matrix (the elements of the traffic matrix are the traffic loads between pairs of nodes in the network). In general, traffic engineering practices assume the knowledge of the traffic pattern, which is not a realistic assumption. As already mentioned, the increasing peer-to-peer traffic and increasing number of servers make it hard to predict the actual traffic distribution. It is much easier to estimate the total incoming/outgoing traffic loads for the network nodes. In [4], [5], Kodialam et al. introduced the two phase routing scheme that uses load balancing. The load balancing allows to express the problem of finding the optimal routing only in terms of the node traffic loads. Although the model developed by Kodialam et al. optimizes balancing coefficients for the general routing protocol, its practicality is critical. Namely, the linear program for the routing optimization is very complex and the optimization lasts very long - in the experiment that we ran it took more than five days even for the smallest of the analyzed real networks. However, the load balancing is a promising technique because it spreads the traffic over the links evenly for all traffic patterns. Load balancing was shown to significantly improve the guaranteed node traffic loads in the networks with regular topologies [1]–[3]. This is a good motivation to explore the performance of practical load balancing schemes.

III. Load Balanced Shortest Path Routing (LB-SPR)

In the proposed routing scheme, the traffic between a node pair (i, j) is routed in two phases. First, portions of the flow from i to j are routed to the intermediate nodes $m \in V$ (V is the set of network nodes). In the next phase, every intermediate node forwards the traffic to its final destination j. The traffic from i to m, and from m to j is routed along the shortest paths. The portion of the flow that is 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. 1 illustrates the case of routing the traffic between the nodes 1 and 5. The first phase of the flow routing is represented by the dashed arrows, and the second phase of the flow routing by the solid ones.

Assume that the network is represented by a directed graph $G = (V, E)$, where V is the set of nodes (vertices) and E is the set of links (edges). The number of nodes in the network will be denoted by N and the number of links by 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*. Here, the outgoing traffic is the traffic that is coming from the customers and going to the network, and incoming traffic is the traffic that comes from the network and is going to the customers.

As mentioned before, when the described load balancing is applied, the link loads depend only on the node traffic loads. Namely, the traffic between any two nodes i and m consists of two components: the traffic b^1_{im}, generated by i and balanced through m, and the traffic b^2_{im}, directed to m and passing through i as the intermediate node. It is easy to see that it holds:

$$b^1_{im} = \sum\nolimits_{j \in V} k_m d_{ij} = k_m s_i \tag{1}$$

$$b^2_{im} = \sum\nolimits_{p \in V} k_i d_{pm} = k_i r_m . \tag{2}$$

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

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

where $F^l_{im} = 1$ if the link l belongs to the shortest path between the nodes i and m, and $F^l_{im} = 0$ otherwise. Obviously, the link load does not depend on the traffic pattern between nodes, but only on the node traffic loads, i.e. traffic loads generated and received by the customers of these nodes.

The problem of the node traffic maximization and the problem of the congestion minimization 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 guaranteed traffic load, since all the initial flows in the network can be increased up to $1/Q$ times, without causing the link overload.

The general-case formulation of the linear program that minimizes congestion has the form:

$$\begin{aligned} &\min Q \\ &\text{(C1)}\ \sum\nolimits_{i=1}^{N} k_i = 1 \\ &\text{(C2)}\ \forall l \in E : \frac{\sum_{(i,m)} F^l_{im}\,(k_i r_m + k_m s_i)}{C^l} \leq Q \\ &\text{(C3)}\ \forall n \in V : \sum_{l \in \text{IN}(n)} L^l - \sum_{l \in \text{OUT}(n)} L^l = r_n - s_n . \end{aligned} \tag{4}$$

Sets $\text{IN}(n)$ and $\text{OUT}(n)$ represent the set of incoming and outgoing links of a node n, respectively. This problem has $N+1$ variables - $k_1, k_2, \ldots, k_N$ and Q. The number of constraints is $M + N + 2$ (not counting non-negativity constraints).

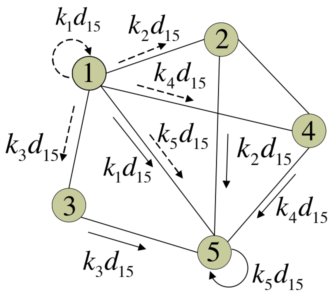

Fig. 1: Routing scheme illustration.

We will prove that if for every node the incoming equals the outgoing traffic, the flow conservation constraints (C3) are superfluous.

Lemma 1: If $r_n = s_n$ the constraint (C3) is redundant.

Proof: When $r_n = s_n$, after substituting L^l from Eq. 3, constraint (C3) becomes

$$\sum_{\substack{(i,m)\\ l \in \text{IN}(n)}} F^l_{im}(k_i s_m + k_m s_i) = \sum_{\substack{(i,m)\\ l \in \text{OUT}(n)}} F^l_{im}(k_i s_m + k_m s_i). \tag{5}$$

For $i \neq n, m \neq n$, node n can either be on the shortest path between i and m or not. If it is not on this path, then there is no link $l \in \text{IN}(n)$ or $l \in \text{OUT}(n)$ such that $F^l_{im} = 1$, and the term $k_i s_m + k_m s_i$ does not appear in the equation. If n is on the shortest path, then there is exactly one link entering and one leaving this node with $F^l_{im} = 1$, so the term $k_i s_m + k_m s_i$ appears on both sides of the equation, and can be eliminated. This leaves only the terms associated with $i = n$ or $m = n$.

Let us now consider the case when $i = n$, i.e. n is the source node. For any m, there can be no link $l \in \text{IN}(n)$ such that $F^l_{im} = F^l_{nm} = 1$ (otherwise the path from n to m would contain a loop, and would not be the shortest path). Similarly, if $m = n$, i.e. n is the destination node, there can be no link $l \in \text{OUT}(n)$, such that $F^l_{im} = F^l_{in} = 1$.

Thus, we can rewrite the Eq. 5 in form:

$$\sum_{\substack{i\\ l \in \text{IN}(n)}} F^l_{in}(k_i s_n + k_n s_i) = \sum_{\substack{m\\ l \in \text{OUT}(n)}} F^l_{nm}(k_n s_m + k_m s_n). \tag{6}$$

If the network is connected, there is a path between every pair of nodes. For any i there is exactly one link $l \in \text{IN}(n)$ such that $F^l_{in} = 1$, so for every i the term $k_i s_n + k_n s_i$ appears exactly once on the left hand side of the Eq. 6. Similarly, for any m there is exactly one link $l \in \text{OUT}(n)$, such that $F^l_{nm} = 1$. For every m, the term $k_n s_m + k_m s_n$ appears exactly once on the right hand side. Equation 6 always holds, because it holds $\sum_{i \in V} k_n s_i = \sum_{m \in V} k_n s_m$ and $\sum_{i \in V} k_i s_n = \sum_{m \in V} k_m s_n = s_n$. Thus, the initial constraint is always fulfilled and is therefore not necessary to add it to the model. ■

According to the previous lemma, we can use the following linear program to minimize the congestion:

$$\begin{aligned} &\min Q \\ &\text{(C1)}\ \sum\nolimits_{i=1}^{N} k_i = 1 \\ &\text{(C2)}\ \forall l \in E : \frac{\sum_{(i,m)} F^l_{im}\,(k_i s_m + k_m s_i)}{C^l} \leq Q \end{aligned} \tag{7}$$

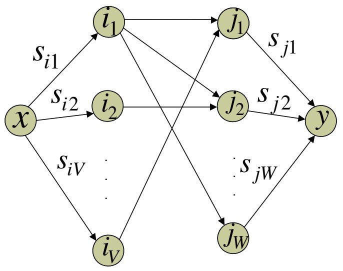

Fig. 2: The assigned graph.

Model (7) now has $N+1$ variables and $M+2$ constraints.

Guaranteed traffic loads that node i can generate and receive are s_i/Q and r_i/Q, respectively. By setting the in-traffic and out-traffic vector elements that are fed into the LP program to be proportional to the desired node weights, the ratio between the guaranteed node loads can be controlled. We will assume that the node weights are proportional to the total link capacities entering and leaving the node. Namely, the capacity of the links for the customers at each node corresponds to the bandwidth demand at each node. Note that this demand is fairly simple to predict. Since the customers at different nodes should be equally served, the incoming and outgoing guaranteed node traffic loads should be proportional to the link capacities allocated to the customers according to their estimated demands. Let the total link capacity for the traffic entering the node i be C_i^{in}, and the total link capacity for the traffic leaving the node be C_i^{out}. So, we choose $s_i \propto C_i^{in}, r_i \propto C_i^{out}, i \in V$. For the considered realistic backbone network topologies, the condition $r_i = s_i$ is fulfilled for every i. That is,

$$r_i = s_i \propto C_i = C_i^{in} = C_i^{out}, i \in V. \tag{8}$$

The guaranteed traffic load for the node i in the network is proportional to the value of the out-traffic vector element s_i, and equals s_i/Q. The minimum value of the guaranteed node traffic load in the whole network equals:

$$s_{gtd}^{lbr} = \min_{i \in V} s_i/Q. \tag{9}$$

IV. Shortest Path Routing

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

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 the set of sources sending their traffic across l by $I^l = \left\{i | \exists j, (i,j) \in P^l\right\}$, and the set of destinations receiving the traffic across l by $J^l = \left\{j | \exists i, (i,j) \in P^l\right\}$. Let us calculate the worst-case traffic load of link l, and then the traffic load that can be guaranteed when SPR is used in a given network. For this purpose, we form a bipartite graph with I^l and J^l as the sets of nodes - Fig. 2. 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 $j \in J^l$ to y, with the capacity $r_j = s_j$. 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 [19]. The worst-case link utilization is $U^l = f_{max}^l/C^l$. For a node i, this implies the node traffic limit on link l to be s_i/U^l. Therefore, the minimum guaranteed node traffic in the whole network equals

$$s_{gtd}^{spr} = \min_{i \in V, l \in E} s_i/U^l. \tag{10}$$

TABLE I: Results for the six Rocketfuel network topologies

NETWORK	N	M	Model Size		G	t[sec]
			var	con		
AS 3967 (Exodus)	79	294	80	282	6.02	2.77
AS 1755 (Ebone)	87	322	88	316	2.02	3.38
AS 1221 (Telstra)	104	302	105	304	2.24	3.53
AS 6461 (Abovenet)	138	744	139	720	4.93	9.36
AS 3257 (Tiscali)	161	656	162	629	5.95	15.92
AS 1239 (Sprintlink)	315	1944	316	1923	7.69	60.36

V. Performance Comparison

We compared the performance of LB-SPR with the performance of SPR on the six backbone network topologies published in the *Rocketfuel* project [15], for the node weight model in (8). We calculated the gain of the guaranteed node traffic loads when balancing is used:

$$G = s_{gtd}^{lbr}/s_{gtd}^{spr}, \tag{11}$$

where s_{gtd}^{lbr} and s_{gtd}^{spr} are defined in (9) and (10). From (7), it follows that load balancing coefficients do not depend on the traffic matrices, they only depend on the link capacities and the estimated node traffic loads. So, if the node traffic loads scale proportionally, load balancing coefficients and gain G do not change.

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. We used the linear program (7) to find the balancing coefficients that maximize guaranteed node traffic loads for the LB-SPR protocol. Then, we calculated the guaranteed node traffic loads in the case of SPR, as described in Section IV, and calculated the gain (11).

The gains and the times needed to optimize the balancing coefficients are given in Table I. Processing time is up to around 1 min which is acceptable in the IP network. Namely, the balancing coefficients are recalculated only when the network topology changes, and the achieved processing times are sufficiently quick to adapt routing to the new network topology in order to utilize it most efficiently. The processing time of the protocol in [5] was more than five days even for Exodus which is the smallest of the analyzed networks. So, long processing time would not be acceptable. The gain of

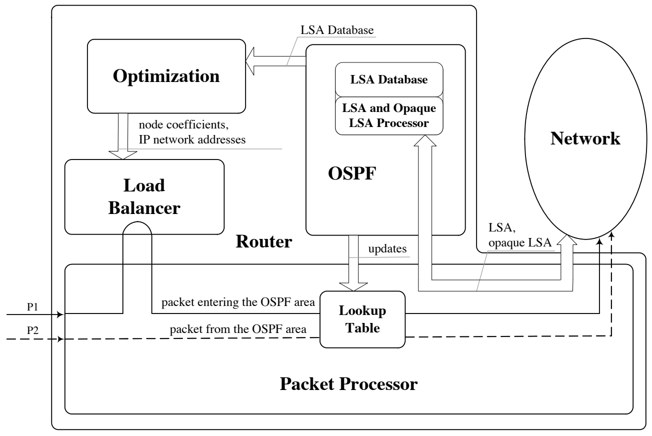

Fig. 3: The scheme of the LB-SPR implementation

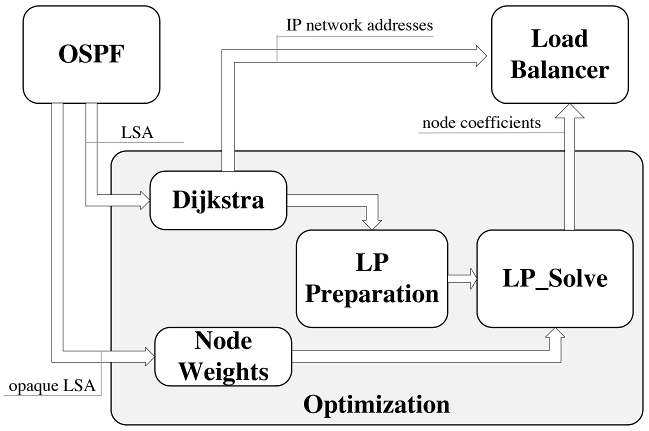

Fig. 4: The scheme of the optimization module

the LB-SPR protocol compared to the commonly used SPR protocol (e.g. OSPF or RIP) depends on the network topology - for the more meshed topologies, the gain values are higher. This was implied by the results of the analysis which was performed for the regular network topologies [3]. Gains for the analyzed realistic topologies are ranging from 2.24 to 7.69, which is quite obviously a significant improvement.

VI. Implementation

In this section, we present the implementation of the previously analyzed LB-SPR routing protocol. In order to make LB-SPR as compatible as possible to OSPF, it is implemented in each OSPF area separately. When a packet enters the OSPF area, its intermediate router is determined. The proposed routing scheme uses OSPF to route the packets between the source router and the intermediate router, as well as between the intermediate router and the destination router. Here, the source router is the first router that the packet encounters when it enters the OSPF area, and the destination router is the last router that the packet passes in the OSPF area under consideration.

In a common IP router that uses the OSPF protocol, when a packet arrives to the router, it is first processed by the packet processor. The packet processor uses its lookup table to determine the router output port to which the packet should be forwarded based on its destination IP address. The lookup table is updated whenever the network topology changes, which provides an autonomic reliability. A software module calculates new lookup table based on the LSA (Link State Advertisement) control packets exchanged through the OSPF protocol, and sends it to the packet processor. The LB-SPR implementation has a similar structure, but it requires the extension of the OSPF module, and some new modules as well. The balancing coefficients are recalculated whenever the network topology changes, which provides the same autonomic reliability as does the OSPF.

The LB-SPR implementation is illustrated in Fig. 3. The solution is based on the OSPF implementation by Moy [17], [18], which is extended to support load balancing. First, it was necessary to allow the retrieval and distribution of the specific information needed by the linear program for the routing optimization, such as the node weights C_i. Also, the new C++ modules were developed to perform the routing optimization and obtain the needed parameters: guaranteed node traffic loads and balancing coefficients k_i. Finally, the load balancer was implemented to route the packets entering the OSPF area according to LB-SPR. Load balancer first has to determine the intermediate router for each incoming packet, and then to direct the packet accordingly. Specified portions of all the flows entering source routers have to be directed to the intermediate routers, according to the calculated optimal values of the coefficients k_i. There are several possible mechanisms to achieve this functionality. We chose the loose source routing as the simplest IP-based solution. Namely, the destination IP address of a packet entering the OSPF area is replaced with the IP address of the intermediate router, while the destination address becomes part of the loose source routing option field.

Let us summarize how the packets are processed in the router shown in Fig. 3. The path for the ”new” packet entering the OSPF area is represented with the full line in Fig. 3. The packet which is entering the OSPF area has to be processed by the load balancer, which determines the intermediate router for the packet, and modifies the IP header accordingly. Once the packet has been modified by the load balancer, it is forwarded through the network using the standard OSPF routing tables. On the other hand, the path of the ”old” packet that has already been modified by its source router is represented by the dashed line. This packet is only passing through the given router, and does not need to be processed by the load balancer. The information needed to route this packet can be obtained from the standard OSPF routing table. In the next subsections, we will describe implemented software modules in more details.

A. Extended OSPF Module

In the case of the regular OSPF, the changes of the network topology trigger the recalculation of the OSPF routes. For LB-SPR, every time the topology changes it is also necessary to repeat the routing optimization and recalculate the balancing coefficients k_i, based on the updated OSPF routing tables.

The node weights C_i are needed to run the optimization (8). These weights can be set by the administrator, or can be, more desirably, autonomic. Therefore, we use the SNMP protocol to detect the operational state of the router interfaces

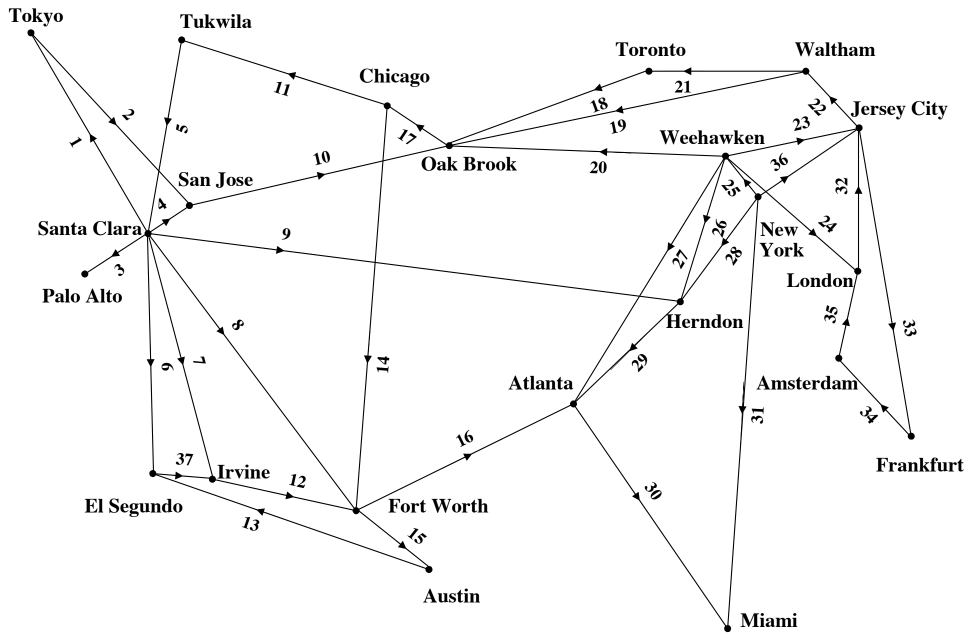

Fig. 5: The analyzed network.

in the network, as well as their speeds. The use of the SNMP to detect the changes of the interface operational states (up or down), and their capacities allow together with the OSPF mechanism full automation of the topology change discovery and distribution.

Using SNMP, each router learns the operational state of its interfaces and their speeds, and distributes this control information inside the OSPF area. The opaque LSAs with the area-local scope are used to convey this information according to the OSPF standard described by RFC 2370 [23]. Opaque LSAs were introduced to provide a generalized mechanism to allow for the future extensibility of OSPF. Opaque LSA consists of the standard LSA header followed by the 32-bit application-specific information field. In our implementation, the opaque type value is selected from the range reserved for experimental and private use. The routers' weights, i.e. external link capacities, are transferred as the 64-bit integer values. Incoming and outgoing opaque LSAs are processed and stored into the LSA database. Whenever the external link capacity changes, the router learns about the change through the SNMP protocol, and distributes the updates by the opaque LSAs.

Standard LSAs carry the information about the network topology. Using this information, the OSPF module calculates the IP routing table and sends this table to the packet processor of the router. Whenever the network topology changes, the OSPF module recalculates the IP routing table and sends its updates to the packet processor. In the LB-SPR implementation, the selected information about the network topology and the capacity of the routers' external (customer) links is transmitted to the optimization module. The OSPF obtains this information from standard LSAs and opaque LSAs. Using this information, the optimization module determines the parameters required to perform load balancing.

B. Optimization Module

The optimization module gets the required information from the OSPF module which performs the signaling, as we have described in the previous subsection. Based on this information, it optimizes the routing based on load balancing, and sends the required parameters to the load balancer which performs the actual routing of incoming packets.

The optimization module is shown in Fig. 4. Based on the network topology information obtained from the OSPF module, the Dijkstra module calculates forwarding trees for all nodes in the network according to the Dijkstra algorithm. The Dijkstra module also calculates the IP network address of each intermediate router through which the traffic will be balanced. This IP address will replace the destination IP address when the source routing is used in the load balancer. Using the calculated trees, the next module in line, the LP preparation module calculates coefficients $F_{ij}^{l}, i, j \in V$ which are required for the linear program defined by (7). Finally, the LP_Solve module optimizes the routing and calculates the balancing coefficients $k_i, i \in V$, which are necessary to the load balancer.

C. Load Balancer

The load balancer receives the balancing coefficients from the optimization module. It also receives the information about the IP network addresses of the intermediate routers. These addresses are calculated by the Dijkstra module, which is the part of the optimization module. The load balancer gets the information that it requires through a TCP connection. Based on this information, the load balancer determines the router output port for each packet entering the router and the OSPF area, and modifies its header in order to balance the traffic appropriately.

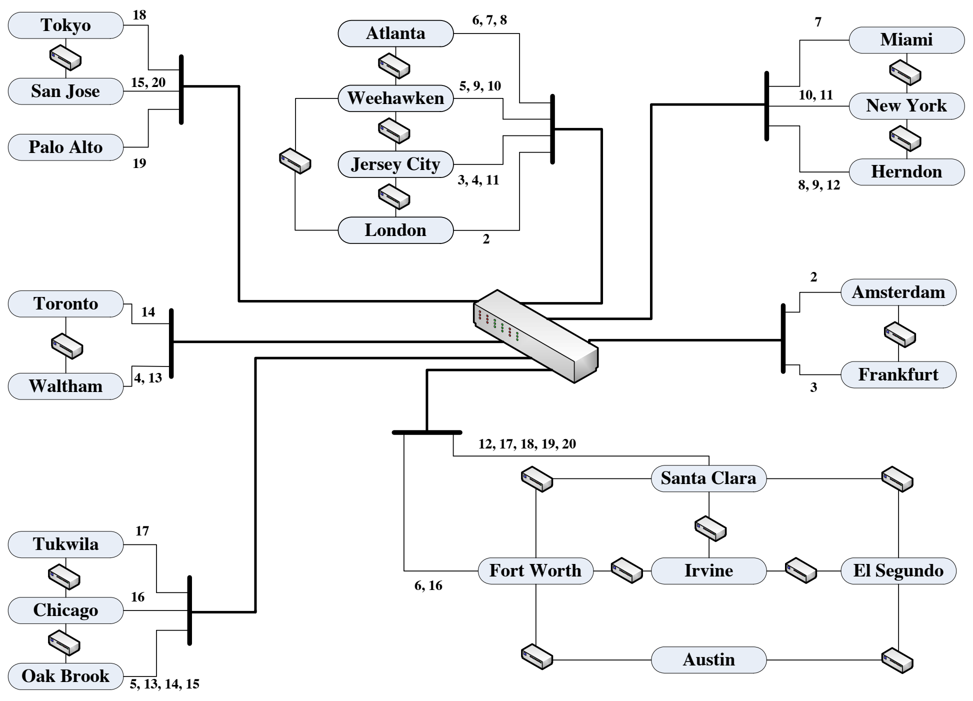

Fig. 6: The simulation environment.

For each destination router j, the load balancer of the given source router i stores the information about the currently used intermediate router m_j. We will call router m_j the active intermediate router. It also maintains a counter with the number of bytes B_j that remain to be balanced across that intermediate router. The initial value of the counter is proportional to the balancing coefficient k_{m_j} of the intermediate router m_j.

When a packet enters the OSPF area, it has to be processed by the load balancer. First, the destination router for the packet is determined, based on the IP address of the packet destination. Let us say that it is a destination router j. Then, the corresponding IP network address of m_j is found, as well as the counter B_j by the search through a Patricia tree. The Patricia tree allows for a fast lookup. Then, the packet header is modified: the destination address is replaced by the IP network address of the intermediate router m_j, and the original destination address is placed in the option field for the loose source routing as defined by RFC 791 [24]. The counter B_j is then decremented by the length of the packet (in bytes). When the counter B_j is smaller than the packet length, the active intermediate router is updated. The next router from the list of possible intermediate routers, $m_j = next(m_j)$, becomes active, and the counter B_j is set to the value proportional to the balancing coefficient corresponding to that intermediate router, k_{m_j}.

The load balancer was developed as a separate program which is executed in the Linux user space. This module uses netfilter/iptables for the acquisition of packets that are entering the OSPF area. The load balancer could be executed by the separate core when a multicore processor is used, if LB-SPR needs to be improved. The speed of the load balancer can be also improved if it is implemented in the Linux kernel, by eliminating unnecessary task switching. Finally, the load balancer can be implemented in hardware of the high-speed ports.

```
ifconfig eth1 100.7.3.2
ifconfig eth1 netmask 255.255.255.0
ethtool -K eth1 tx off
ifconfig eth2 100.7.2.2
ifconfig eth2 netmask 255.255.255.0
ethtool -K eth2 tx off
ifconfig eth3 200.0.22.1
ifconfig eth3 netmask 255.255.255.0
ethtool -K eth3 tx off
mount 192.168.122.1:/root/router /root/tmp
for f in /proc/sys/net/ipv4/conf/*/accept_source_route
do echo 1 > $f
done
```

Fig. 7: Configuration file of the router in Amsterdam

VII. Testing

A. Testing Environment

The network simulation using virtual machines as the network nodes is gaining more popularity with the increase of the computer processing power. The advantage of this simulation method is that the protocol under consideration is executed as it would be executed in the exploitation conditions. This way, the more accurate simulations are performed, and possible errors when adapting the protocol implementation to the simulation tools are avoided.

We selected Xen as the virtualization platform because of its maturity and performance [22]. Xen allows several operating systems to be executed on the same computer hardware concurrently. Administrator starts these operating systems through the main operating system, named *dom0*,

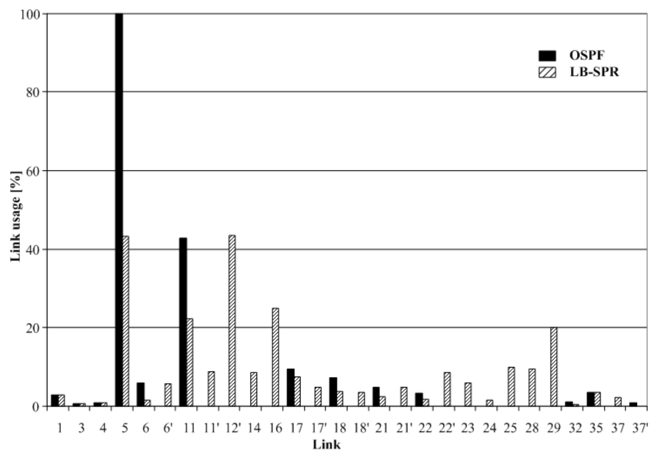

Fig. 8: The link usage.

which boots automatically. Existing configuration tools enable effortless configuration of the virtual machines and their simulation. These virtual machines, called *domU*, will be connected through the bridges created in the *dom0* kernel. The network interfaces will be created manually by using the configuration shell scripts. The virtual machines implement the entire TCP/IP stack in which the OSPF protocol will be replaced by the LB-SPR protocol shown in Fig. 3. The virtual machines will play the role of both routers and hosts in the network. The data will be exchanged in the network, and we will test the LB-SPR functioning using the Multi Router Traffic Grapher (MRTG) packet analyzer [25].

B. Functional verification of the LB-SPR Implementation

The performance of LB-SPR was analyzed in the network represented in Fig. 5. This is, in fact, the simplified version of the Exodus network topology, published by the Rocketfuel [15]. For the purpose of this simulation, all the nodes in one city were represented by a single node, and the equivalent link weights were calculated. This network was emulated using seven computers and one Ethernet switch as represented in Fig. 6. Depending on the processor speed and RAM size, the number of the virtual routers executed on a single computer ranges from two to five. The virtual routers X and Y on a single computer are connected through the Xen bridge xenbrXY. The links between the virtual routers on different computers were established using VLANs. The VLAN tags corresponding to the links between routers executed on different computers are represented by the numbers in Fig. 6.

Each virtual router is configured using the configuration script. A representative configuration file of the router in Amsterdam is shown in Fig. 7. In this configuration file, the IP addresses and the corresponding masks are assigned to the network interfaces, TCP checksum offload is disabled and a loose source routing is allowed. Communication channel bandwidths are configured using Linux traffic control tool, tc.

For the analyzed network, the worst-case traffic pattern for OSPF was determined using the maximum matching algorithm. The critical link for OSPF is the link between Tukwila and Santa Clara. It gets congested when the following pairs of nodes communicate with the maximum speeds: Oak Brook - San Jose, Toronto - Palo Alto, Amsterdam - Santa Clara, Tukwila - Irvine, Chicago - Tokyo, and Waltham - El Segundo. The traffic between these nodes was set to the value that causes the critical link utilization to be 100%. Then, the LB-SPR is applied for the same traffic pattern. The link utilizations for both OSPF and LB-SPR are observed using the MRGT packet analyzer and plotted in Fig. 8. The link utilizations in the case of OSPF are represented by the black lines, and in the case of LB-SPR by the gray ones. The link numbers in Fig.5 and Fig. 8 match. In Fig. 8, l' denotes the link connecting the same nodes as the link l, but in the opposite direction. Only the results for the used links are plotted. It can be observed that the OSPF protocol uses 13 links to route the given traffic pattern, while the LB-SPR protocol uses 27 links. At the same time, the congestion of the critical link is lowered to 43.3% when LB-SPR is applied. Simulation results agreed with the analytical results which confirmed the correctness of the implementation. They show that LB-SPR balances the traffic over the network links better than OSPF. If the critical link 5 had two times lower capacity, LB-SPR would still pass the traffic, while SPR would not.

VIII. Conclusion

In this paper, we presented a novel routing protocol for the IP network, which is based on load balancing. This protocol is automated as the existing routing protocols such as OSPF, and adapts to the changes of the network topology. LB-SPR calculates the traffic loads that the nodes can guarantee to carry. Using the information about the guaranteed node traffic loads, the bandwidth reservations become simple in such a network, and, consequently can be made fast. Fast and autonomic bandwidth reservations are important for the multimedia applications whose popularity is growing. At the same time, the LB-SPR protocol maximizes the node traffic loads that can be guaranteed in the given network. It was shown that LB-SPR improves the guaranteed traffic up to 7.7 times for the real networks that we considered, compared to the shortest path routing protocols such as OSPF. Our analysis showed that the gain of LB-SPR increases with the average node degree, which is in agreement with the results for the regular network topologies. We also implemented the LB-SPR protocol using existing IETF standards, and integrated this protocol into the TCP/IP stack. Virtual machines on multiple computers were used to simulate and validate the network that runs the implemented LB-SPR routing protocol.

Since LB-SPR is using the OSPF signaling, it inherits its recovery speed which is insufficiently low for the interactive applications. If a faster recovery mechanism is needed, it can be employed at the lower layers as it is typically done. Alternatively, the capacities can be overprovisioned to account for the failures. We plan to compare the costs of the networks using LB-SPR and OSPF in which the link capacities are overprovisioned to pass given node traffic loads even when single failures, of nodes or links, occur. Whenever the topology changes, the balancing coefficients would be optimized to utilize the network most efficiently. Also, we plan to implement described bandwidth reservation mechanism using the IETF protocols such as RSVP or SIP together with LB-SPR as the underlying routing protocol.

References

[1] A. Smiljanić, "Rate and Delay Guarantees Provided by Clos Packet Switches with Load Balancing", *IEEE Trans. Netw.*, Feb. 2008.

[2] I. Keslassy, C. S. Chang, N. McKeown, D. S. Lee, "Optimal Load-Balancing", in *Proc. INFOCOM 2005*, March 2005.

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

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

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

[6] M. Antić, A. Smiljanić, "Routing with Load Balancing: Increasing the Guaranteed Node Traffics", *IEEE Commun. Lett.*, June 2009.

[7] N. Maksić, P. Knežević, M. Antić, A. Smiljanić, "On the Performance of the Load Balanced Shortest Path Routing", *Proc. PacRim09*, 2009.

[8] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An Overview of Routing Optimization for Internet Traffic Engineering", *IEEE Commun. Surveys & Tutorials*, 1st Quarter 2008.

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

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

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

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

[13] 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.

[14] R. Cohen and G. Nakibli, "On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing"', *IEEE Trans. Netw.*, June 2008.

[15] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring Link Weights Using End-to-End Measurements," *Proc. of the IMW 2002*, Nov. 2002.

[16] *LpSolve, Reference Guide* [Online]. Available:http://lpsolve.sourceforge.net, 2005.

[17] John T. Moy, *OSPF Complete Implementation*, Addison-Wesley Professional, 2000.

[18] *OSPFD Routing Software Resources* [Online]. Available:www.ospf.org

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

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

[21] B. Towles and W. J. Dally, "Worst-case Traffic for Oblivious Routing Functions," *Computer Architecture Letters*, 1, Feb. 2002.

[22] *Xen Users' Manual, Xen V2.0 for x86*, [Online]. Available: http://www.xen.org/files/xen_user_manual.pdf

[23] *RFC2370: The OSPF Opaque LSA Option*, [Online]. Available:http://www.ietf.org/rfc/rfc2370.txt

[24] *RFC791: Internet protocol*, [Online]. Available:http://www.ietf.org/rfc/rfc791.txt

[25] *MRTG 2.16.2 configuration reference*, [Online]. Available:http://oss.oetiker.ch/mrtg/

Marija Antić received her B.Sc. and M.Sc. degrees in electrical engineering from Belgrade University in 2005 and 2008, respectively. Currently she works as a Research Assistant at Belgrade University. Her area of research are communication protocols and optimization. Marija was among the best students in her class and received various awards and scholarships, including the German DAAD grant and the Norway Government Award for the best students of Belgrade University. Her work on the master thesis was supported by the Serbian Ministry of Science. From 1997 until 2007 she received the scholarship of the Serbian Ministry of Education, for her academic record and outstanding results in mathematics and physics competitions.

Nataša Maksić received her B.Sc. degree in electrical engineering from Belgrade University in 2007, as the best student in her class, with the maximum average grade of 10. Currently, she works as a Research Assistant at Belgrade University. Her research interests are communication networks and protocols. Nataša was awarded a scholarship by the Serbian Ministry of Education from 2003 until 2005 for her academic record, and got several awards from the companies such as YUBC systems and Eurobank EFG.

Petar Knežević received B.Sc. and M.Sc. in electrical engineering from Belgrade University in 2000 and 2006, respectively. Since 2000, he works as a software developer at Iritel, Belgrade. Areas of his research interest are communication protocols and programming. Petar published numerous papers in the area of communication protocols and their implementation. He works in the area of the network management protocols. In particular, Petar works on the development of the multiservice SDH equipment.

Aleksandra Smiljanić (M '96) received the M.A. and Ph.D. degrees in electrical engineering from Princeton University in 1996 and 1999, respectively. She got the B.Sc. degree in electrical engineering at Belgrade University in 1993. Her area of research is high performance switching and routing. Currently, Aleksandra works as a Professor at Belgrade University in Serbia, and as an Associate Research Professor at Polytechnic Institute of New York University. She had worked for AT&T Labs Research from 1999 until 2004.

Aleksandra Smiljanić is the author of numerous conference and journal papers in the area of high performance switching and routing. She is the inventor of nine US patents. Aleksandra Smiljanić is the author of the Best Papers at IEEE Conference on High Performance Switching and Routing 2000 and 2002. She got the Research Excellence Award at AT&T Labs in 2000. Aleksandra Smiljanić is the Editor of IEEE Communication Letters and of OSA Journal on Optical Networking.