Improving Utilization of Data Center Networks

Nataša Maksić and Aleksandra Smiljanić, Belgrade University

ABSTRACT

In this article, we analyze and compare different data center environments in terms of the maximal throughput they offer to end nodes. In particular, we examine to which extent balancing can increase the throughput guaranteed for an arbitrary traffic pattern. We consider and compare two-phase balancing algorithms which are optimized assuming different underlying routing algorithms. It will be shown that optimized load balancing significantly improves the maximal loss-free throughput. A suite of routing algorithms is examined on three cost-effective data center topologies: fat-tree, dragonfly, and flattened butterfly.

INTRODUCTION

Data centers are becoming Internet brains with thousands of servers and switches. They process and exchange enormous amounts of data at high speeds in a limited space. To go beyond the speed of a single processing node, very fast and efficient networking is required. In addition, packet loss should be minimized in data centers, since many applications are delay-sensitive, and retransmitted packets would be too late for them.

The goal of reaching maximal guaranteed throughput is imperative for being competitive on the market. In data center topologies, which have many alternative paths, this goal can be achieved through load balancing. Load balancing eliminates adversial traffic patterns that could lead to link overloads by evenly distributing the traffic across network links. Load balancing can be optimized for maximal guaranteed traffic using linear programming [1, 2]. We present analytical and experimental comparisons of the selected routing algorithms, and their extensions with optimized load balancing. It is shown that the highest throughput in a loss-free network is achieved by an optimized load balancing scheme based on equal-cost multipath routing, which is adapted to a plethora of equal-cost paths typical for data center networks.

This article is organized as follows. The second section briefly introduces data center topologies with emphasis on fat-tree, flattened butterfly, and dragonfly topologies, which are assumed in our analysis. The third section describes routing algorithms which are analyzed. The fourth section presents analytical results for the maximal guaranteed traffic loads in data center networks supported by different routing algorithms, as well as simulation results. The article is concluded in the fifth section.

DATA CENTER TOPOLOGIES

Many different topologies have been proposed in the literature, and a few of them are used in data centers. In practice, the topologies that minimize cost are naturally preferred, and they are typically based on cheap commodity switches. The cost of enterprise switches is low due to economy of scale; however, their hardware is not particularly designed for data center networking. For this reason, in some of the proposed topologies, servers play the role of routers in order to provide flexibility while keeping the cost low. However, servers cannot be as scalable as routers, which are specialized equipment.

The typical topology of data center networks used to be hierarchical, usually with three layers of hierarchy. The switches at higher layers ought to be large in such a topology, and they ultimately become a bottleneck of data centers, which need to host thousands of servers. In addition, cost per bit per second is significantly higher for higher-capacity switches. For these reasons, data centers with regular topologies are becoming popular, in which switches of the same or similar sizes are used. The most prominent regular topologies proposed for data center networking are fat-tree, dragonfly, flattened butterfly, BCube, and DCell [3–6]. Toroidal and hypercube topologies are used in parallel computing clusters, because they are very easy to grow as nodes are connected to neighbors. On the other side, torus and hypercube have been shown to incur much higher cost for the same throughput than the more advanced topologies we consider: fat-tree, dragonfly, and flattened butterfly [3, 5, 6].

It has been shown that advanced techniques such as load balancing and adaptive routing can significantly improve performance of data centers [5–7]. Cheap commodity switches commonly used in data centers are inflexible and cannot support these novel mechanisms. For this rea-

This work was funded by the Serbian Ministry of Science, and companies Telekom Srbija and Informatika.

0163-6804/13/$25.00 © 2013 IEEE

son, in several papers it has been proposed that servers should play the role of routers besides running applications and generating traffic, as in DCell or BCube architectures [4]. That is, servers could implement arbitrary routing policies and improve network performance. However, packet processing is a heavy burden for general-purpose processors, and consequently, using servers instead of specialized networking equipment is less scalable.

In this article, we examine novel routing protocols in fat-tree, flattened butterfly, and dragonfly topologies because they are the most scalable and cost effective.

Fat-Tree Topology

Fat-tree or folded Clos topology has been recognized as a good solution to replace hierarchical data centers because of their better scalability and lower cost [3, 7]. Clos networks were initially proposed for building high-capacity circuit switches in telephone networks using smaller switching elements. Later, it was recognized that high-capacity packet switches can be built in the same way. In Clos packet switches, there is no time to find a path for every single packet through the network; therefore, load balancing is used for efficient switch utilization [8, 9]. Nowadays, folded Clos or fat-tree topology is used in data centers as well.

In the most popular three-stage Clos network, switches in each stage are connected to all switches in the following stage. In folded Clos, switches in the first and third stages of an initial Clos network are merged into the edge switches, which are connected to the top-level switches belonging to the second stage of the initial Clos network. In general, a $(2k, n)$ fat-tree network has n stages and k^2 top-level $k \times k$ switches. Edge switches are built recursively; an edge switch of the $(2k, n)$ fat-tree network is actually the $(2k, n-1)$ fat-tree network in which top-level switches have additional ports toward the top-level switches of the parent $(2k, n)$ fat-tree network. An example of a $(2k, 3)$ fat-tree network is given in Fig. 1. It can be observed in Fig. 1 that a fat-tree network provides many different paths between any pair of servers.

Flattened Butterfly Topology

In [3], it was assumed that ports represent the main part of the switch cost, and it was shown that hierarchical networks become much more costly than fat-tree networks as the number of servers exceeds a couple of thousand. In [5, 6], it is conversely argued that links, including their ports, represent the main part of the switch cost. For this reason, flattened butterfly topology is proposed as it decreases the number of links in the network compared to the fat-tree topology.

Flattened butterfly topology is derived from a butterfly self-routing network in which subsequent segments of the packet address determine the output port of the switch at each stage. In general, a k-ary n-fly butterfly network comprises n stages with $N/2$ $k \times k$ switches in each stage. Switches of the first stage are connected to switches 0, N/k, $2N/k$, …, $(k-1) \cdot N/k$ in the second stage, switches in the ith stage are connected to switches 0, N/k^i, $2N/k^i$, …, $(k-1) \cdot N/k^i$ in the next stage, and so on. Flattened butterfly topology is obtained from a butterfly network by merging switches with the same ordinal number in all stages.

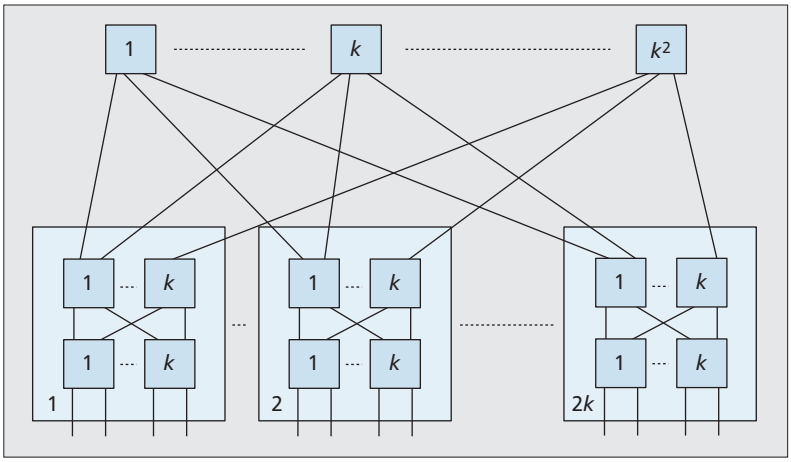

Figure 1. *Fat-tree topology.*

The flattened butterfly topology is shown in Fig. 2, in which switches are identified by ordinal numbers written in binary format. Each link is denoted by the binary number in which symbol x can be replaced by digits 0 or 1 to produce identifiers of the switches connected to that link. As we can observe in Fig. 2, the flattened butterfly topology has a smaller number of links compared to the fat-tree topology, but it still offers a large number of shortest paths between servers.

Dragonfly Topology

Dragonfly topology was specially tailored to decrease the cost of data centers [6]. It has been observed that the cost of links in data centers increase linearly with their length. Cost per unit distance is lower for optical cables, while their fixed cost is higher than that of electrical cables. Optical cables are more cost effective for links longer than 10 m. The motivation behind the dragonfly topology is to decrease the data center cost by reducing the number of global links with lengths exceeding a specified threshold.

Dragonfly topology is presented in Fig. 3. All switches are divided into groups with a switches per group. In each switch, h ports are connected to the switches in other groups, and the remaining ports are connected to other switches of the same group or to the servers. Switches of each group are connected to the switches in all other groups. Most commonly, each switch is connected to all other switches in the group, as shown in Fig. 3. Alternatively, interconnection of the switches in a group can be based on the flattened butterfly topology.

Dragonfly topology is based on the assumption that the cost of the switch hardware is negligible, which is true in commodity switches with small buffers and minimal packet processing capabilities. However, performance of data centers could be improved with more advanced switching and routing technologies, which would obviously have a higher cost. In that case, the assumption that global links represent the major part of data center cost should be revisited.

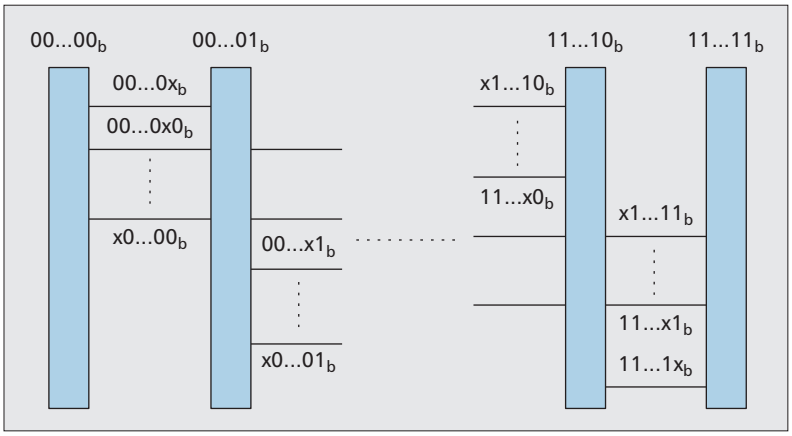

Figure 2. *Flattened butterfly topology.*

Also, dragonfly topology could provide savings if the traffic is localized in nature, and applications need to be aware of the network topology.

Routing Protocols

Routing protocols determine the paths that packets take from their sources to their destinations. Shortest path routing (SPR) algorithms are the most commonly implemented in networks. However, they are being replaced in data center networks by routing algorithms based on load balancing, which more evenly utilize regular topologies of data centers. Valiant balancing is often considered, but it is defined in different ways. According to the most common definition of the Valiant balancing algorithm, a server sends each packet with equal probability through any router in the network.

The best, most even load balancing is achieved when a path is determined for each packet separately. In that case, packets of the same flow could be reordered because they are taking different paths, incurring different delays. Alternatively, flows between some source-destination pair can be sent along different paths so that the packets belonging to the same flow will follow the same path. Such load balancing of flows achieves significant improvements of network utilization compared to shortest path routing. On the other hand, flow-based load balancing is imperfect and unpredictable since flows can have different sizes, which are hard to anticipate at their starting times when they are assigned to the balancing paths.

Hedera [10] and multipath TCP [11] are two recent solutions for the problem of large flows that disrupt even traffic distribution in the flow-based balancing algorithms. Both algorithms monitor flow sizes and react to the occurrence of large flows. Since flow-based load balancing requires additional complexity for monitoring and assignment of the traffic flows, we consider packet-based load balancing algorithms in this article. Also, reordering of packets can easily be resolved at the transport or application layer.

Adaptive routing has also been reconsidered for data centers [5, 6]. In adaptive routing, the paths that packets take depend on the traffic load. Packets usually take a path based on the buffer occupancy of the downstream switches that are on the shortest paths toward the destination. Adaptive routing is risky because it causes instabilities in the system. Also, their interaction with reactive transport protocols such as TCP would be very hard to predict.

In this article, we examine only oblivious routing protocols that do not adapt to the traffic load. We optimize load balancing for different underlying routing algorithms: SPR, SPR with minimal node degree (SPRm), and equal cost multipath (ECMP).

Shortest Path Routing

The most commonly used routing protocols in wide area networks are shortest path routing (SPR) protocols, such as Intermediate Server to Intermediate Server (IS-IS), Open Shortest Path First (OSPF), and Routing Information Protocol (RIP). The Border Gateway Protocol (BGP) also routes packets through different domains along shortest paths, but it considers policies introduced by domain operators as well. The shortest path between two nodes is the one comprising links with the smallest sum of weights. Link weights are typically proportional to the transmission cost through the respective links.

Obviously, SPR protocols are the most efficient since they utilize the network resources with the lowest cost. However, the situation might arise in which some links get congested while others remain underutilized when SPR protocols are used. SPR protocols are not well suited for data centers, because they do not make advantage of the path diversity provided by the regular topologies of data centers. We compare the performance of SPR protocols with the performance of routing protocols based on load balancing.

Shortest Path Routing Using Minimal Node Degree

In the Dijkstra algorithm commonly used in SPR algorithms [12], a parent of a new node is the neighboring node with the minimal ID, which is selected among the nodes on the tree with equal costs of the paths to the root. In data center topologies, which contain multiple equal-cost paths, such candidate node selection is not likely to utilize available paths to the fullest. This would, in turn, create nodes with a large degree in the tree, which would result in concentration of traffic toward the tree root. If a node has large degrees in multiple trees, it would be clogged even for uniform traffic demand across servers.

In order to reduce node degrees in Dijkstra trees and provide alternative SPR paths for evaluation presented in this article, we propose the SPRm algorithm. When a new node is added to the Dijkstra tree in SPRm, its parent is selected among the neighbors in the network that are on the tree and connected to the root node through the paths with minimal costs. Among these candidates, the neighboring node with the minimal number of children is selected in our proposed SPRm. If there are multiple such nodes, the one with the minimal ID is selected.

SPRm maintains low complexity and high

execution speed, and improves the throughput achieved by SPR in regular networks if there are multiple equal cost routes between endpoints.

Equal Cost Multipath Routing Algorithm

Equal cost multipath routes the traffic along all existing equal cost paths. Each router keeps a routing table of all equal cost paths toward destinations in a network and routes packets heading to a given destination balanced across the appropriate links. Besides resilience against node and link failures, ECMP provides more balanced traffic distribution in the network, and higher network throughput for typical traffic patterns.

The ECMP algorithm can be defined to balance either packets or flows. In the first case, each router determines the next hop for a packet based on its destination and the previous utilization of the equal cost paths toward this destination. In the second case, the path is determined in the same way for the first packet of a flow, while the subsequent packets of the flow follow the same path. The advantage of flow balancing is that the packets are delivered in order. However, balancing is imperfect since different flows have different sizes. For more even balancing, flow monitoring and assignment to the paths must be implemented, which increases the routing complexity. For these reasons, we analyze packet-based ECMP due to its simplicity and better performance.

Valiant Balanced Routing

Valiant balanced routing (VBR) was proposed a long time ago in the context of parallel processing [13]. Valiant proposed to route each packet entering the network through a randomly selected intermediate node. In the particular network under observation, a packet was sent to this intermediate node along the shortest path, and from the intermediate node to the final destination again along the shortest path.

In the literature, VBR balancing was defined in different ways depending on the topology, and particular definitions were chosen in an ad hoc manner based on intuition [5, 7]. We adopt the original definition of VBR in which each packet is transferred through an intermediate node that is selected randomly out of all nodes in the data center network.

Optimized Load Balanced Routing Algorithms

In our previous work [1, 2, 14], we proposed load balanced shortest path routing (LB-SPR) for wide area networks. LB-SPR is similar to Valiant balancing, but it optimizes balancing coefficients through different network nodes so that the network throughput is maximized.

For each node in the network, a balancing coefficient is defined as the portion of packets that should be balanced through that node. In Valiant balancing, all balancing coefficients are equal. In the load balancing we are proposing and analyzing, values of these coefficients are optimized assuming a specific topology and the weights assigned to the network nodes according to their capacities. Balancing coefficients are optimized so that the network throughput is maximized while the guaranteed node traffic loads are proportional to their respective weights.

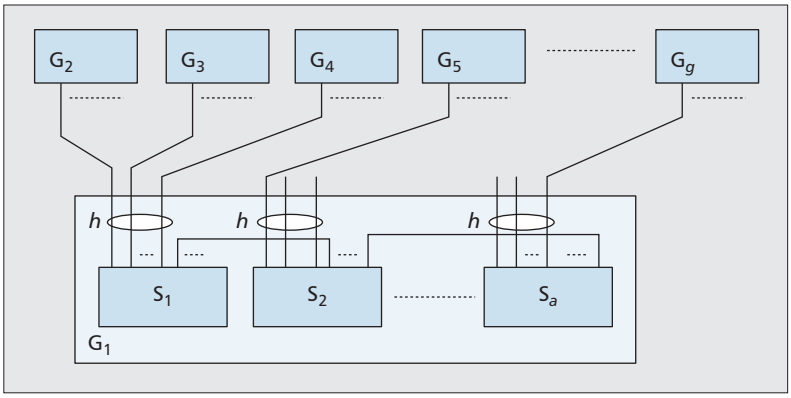

Figure 3. *Dragonfly topology.*

Optimized load balancing can utilize various underlying routing algorithms besides conventional SPR. An underlying routing algorithm determines the path of a packet from the source node to the intermediate node, and from the intermediate node to the destination node. In this article, optimized load balancing assumes SPR, SPRm, and ECMP as the underlying routing algorithms, described in the previous sections. We introduce two novel routing algorithms: load balanced SPRm (LB-SPRm, and load balanced equal cost routing (LB-ECR).

Balancing coefficients k_i, $0 \le i \le N-1$ are optimized using the linear program that minimizes the maximal link utilization, Q, in the network, and therefore maximizes the network throughput. In the LP model, the sum of contributions from each flow to the traffic of a particular link should be smaller than the maximal link utilization in the network, Q. A loss-free network is achieved if the maximal link utilization is smaller than 1. In addition, the sum of balancing coefficients, k_i, in the model should be equal to 1.

Due to the two-stage load balancing, link loads only depend on balancing coefficients k_i, generated traffic loads s_i, and terminated traffic loads r_i of all switches $0 \le i \le N-1$ (s_i and r_i are proportional to the node weights). The linear program maximizes Q under the following constraints:

$$\begin{aligned} & Q < 1 \\ & \forall l : \frac{\sum_{(i,m)} F_{im}^{l}(k_i s_m + k_m r_i)}{C^{l}} \le Q \\ & \sum_{i=1}^{N} k_i = 1 \end{aligned} \tag{1}$$

Factor F_{im} in Eq. 1 represents the contribution of the node pair (i, m) to the link load, where one of those nodes is the intermediate node and the other is either the source or the destination. The F factors are calculated according to the underlying routing algorithm. In SPR

The number of alternative paths with the same cost in dragonfly topology is relatively small. As a result, the advantage of ECMP with respect to the SPR routing, as well as the advantage of LB-ECR compared to LB-SPR, are not as pronounced as in the flattened butterfly network.

Routing Algorithm	SPR	LB-SPR	SPRm	LB-SPRm	ECMP	LB-ECR	VBR
Fat-tree	5	125	25	125	125	125	89.285
Dragonfly	11.333	69.515	11.333	72.009	17	76.81	60.176
Flattened butterfly	5.33	40.58	4.571	43.63	29.538	64	64

Table 1. *Maximal normalized network offered load with no losses.*

and SPRm, a link can be used by the traffic between a node pair or not, so $F_{im} \in \{0, 1\}$; in ECMP, this traffic can be split in nodes along the path; as only its portion might be transmitted through the link in that case, F_{im} could be less than 1. Solution of the described linear model provides values of Q and balancing coefficients. The guaranteed traffic load for node i is calculated as s_i/Q.

Optimized load balanced routing algorithms (LB-SPR, LB-SPRm, LB-ECR) can be defined to balance either flows or packets. In the first case, the flow bit rates must be learned, and the flows need to be assigned to the intermediate nodes according to the balancing coefficients assigned to these nodes. As mentioned before, flow-based balancing delivers packets in order, but spreads the traffic unevenly across the network links. More even spreading can be achieved at the expense of increased implementational complexity. For these reasons, we analyze LB-SPR, LB-SPRm, and LB-ECR based on packet-by-packet balancing.

Analysis and Experimental Results

We calculate the maximum throughputs guaranteed in loss-free networks by the routing algorithms introduced before: SPR, LB-SPR, SPRm, LB-SPRm, ECMP, LB-ECR, and Valiant balancing. These algorithms are analyzed in fat-tree, flattened butterfly and dragonfly data center topologies. Then, we will simulate the performance of the routing algorithms under the consideration in lossy networks with the same topologies. The parameters of the network topologies are selected to have similar bisection bandwidths.

In the analysis, (10,3) fat-tree topology is used with 10-port switches and three levels, having a bisection bandwidth of 62.5 Gb/s. Flattened butterfly topology has a dimension $d = 6$, and dragonfly topology has $g = 17$ groups and $a = 4$ switches in a group, with $h = 4$ connections from each switch toward other groups. Consequently, flattened butterfly and dragonfly topologies have bisection bandwidths of 64 Gb/s and 68 Gb/s, respectively. Analyzed network topologies comprise similar numbers of switches of similar sizes. In particular, the fat-tree network under consideration comprises 75 10-port switches and 125 servers, the flattened butterfly comprises 68 9-port switches and 136 servers, while the dragonfly network comprises 64 8-port switches and 128 servers. Link capacities are 1 Gb/s.

Reference [15] presents a method for finding a set of communicating node pairs that will cause the worst case link utilization in a given network for the specified routing algorithm. For each link in the network, a bipartite graph is generated with all communicating nodes in both partitions. An edge between two nodes is assigned a weight according to the traffic load these two nodes contribute to the link. The worst case traffic pattern for that link is found using the maximum weight matching of the graph.

The worst case traffic pattern for the network includes the communicating pairs that cause the maximal utilization of the most heavily loaded link. Table 1 shows the maximal offered load with no losses, normalized to the link capacity, for the analyzed topologies and routing schemes assuming the worst case traffic pattern.

We have also conducted an evaluation of the discussed routing protocols using the ns-3 network simulator. We have implemented a generator of network topologies, and performed experiments using fat-tree, flattened butterfly, and dragonfly topologies. The worst case traffic patterns are assumed, which are calculated as in [15].

Network throughput for constant bit rate UDP traffic is depicted in Fig. 4. Figure 4 presents results for SPR, SPRm, LB-SPR, LB-SPRm, ECMP, Valiant, and LB-ECR routing. The graphs present the total network throughput, which increases linearly with the offered load when there is no packet loss or packet loss is limited. As packet loss increases, throughput rises more slowly, reaches saturation, and in some cases even declines. Table 1 presents the offered traffic loads for which the links get overloaded and packet losses begin.

From Fig. 4 and Table 1, one can observe that balancing provides higher throughputs for the worst case traffic patterns due to the more even distribution of traffic across network links. LB-ECR performs well in all topologies, and provides the highest guaranteed throughputs in loss-free networks, as can be observed from Table 1. LB-ECR achieves two times higher maximum throughput for the zero packet loss than standard ECMP in the flattened butterfly network and four times higher throughput in the dragonfly network. In fat-tree networks, LB-ECR and ECMP perform similarly. Figure 4 shows that ECMP exhibits very good performance for higher packet losses. However, applications in data centers are typically delay-sensitive and do not tolerate retransmissions, which is why packet losses should be minimized.

For fat-tree topology, all balancing algorithms perform well, and the highest throughputs are

achieved. This is a consequence of the constant bandwidth between levels and the fact that all paths across any level have equal costs. Only Valiant balancing is slightly worse because it unnecessarily detours traffic through the topology when a balancing node is not on the path of a packet. We can also observe that SPRm significantly outperforms the conventional SPR algorithm while maintaining packets in order. On the other side, fat-tree topology requires more switches with a larger number of ports compared to the other topologies.

In the butterfly topology, LB-ECR provides a theoretical maximum for the guaranteed traffic loads. In this case, balancing coefficients are equal, and LB-ECR is equivalent to the Valiant balancing.

The number of alternative paths with the same cost in the dragonfly topology is relatively small. As a result, the advantage of ECMP with respect to the SPR routing, as well as the advantage of LB-ECR compared to LB-SPR, are not as pronounced as in the flattened butterfly network. For the same reason, SPRm does not outperform SPR.

LB-SPR, LB-SPRm, and LB-ECR achieve balanced distribution of traffic across links in the network. As the network throughput increases, most links become overloaded with similar amounts of traffic, and packet drops occur on all those links. With full queues, traffic toward the intermediate nodes, and traffic from the intermediate nodes toward the destinations contest for the available bandwidth. This results in a decline of the total throughput in a lossy network, because the flows toward the intermediate nodes increase with the increase of the offered traffic, and they overtake resources from the flows heading toward their destinations.

Conclusion

We have calculated maximal guaranteed network throughputs when promising routing schemes are used in typical data center topologies, and confirmed theoretical results through experimental measurements. It has been shown that the optimization of load balancing improves guaranteed throughput in a loss-free network. In particular, the proposed LB-ECR algorithm achieves the highest guaranteed throughputs in loss-free networks. Loss-free networks are well suited to data center applications, which are typically delay-sensitive and do not tolerate retransmissions. The underlying routing algorithm of LB-ECR, ECMP, was shown to perform very well in lossy networks compared to the other algorithms.

In LB-ECR, LB-SPR, and LB-SPRm, we optimize load-balancing coefficients using the simple LP model ,which eliminates the flow list or traffic matrix. The traffic matrix and flow list are eliminated by routing every packet via a balancing router, thus excluding direct connection of endpoints in the linear model. Our method uses the traffic vector, containing the total input and output traffic load of each node. This eliminates the need for predicting complex behavior of applications typical for data centers.

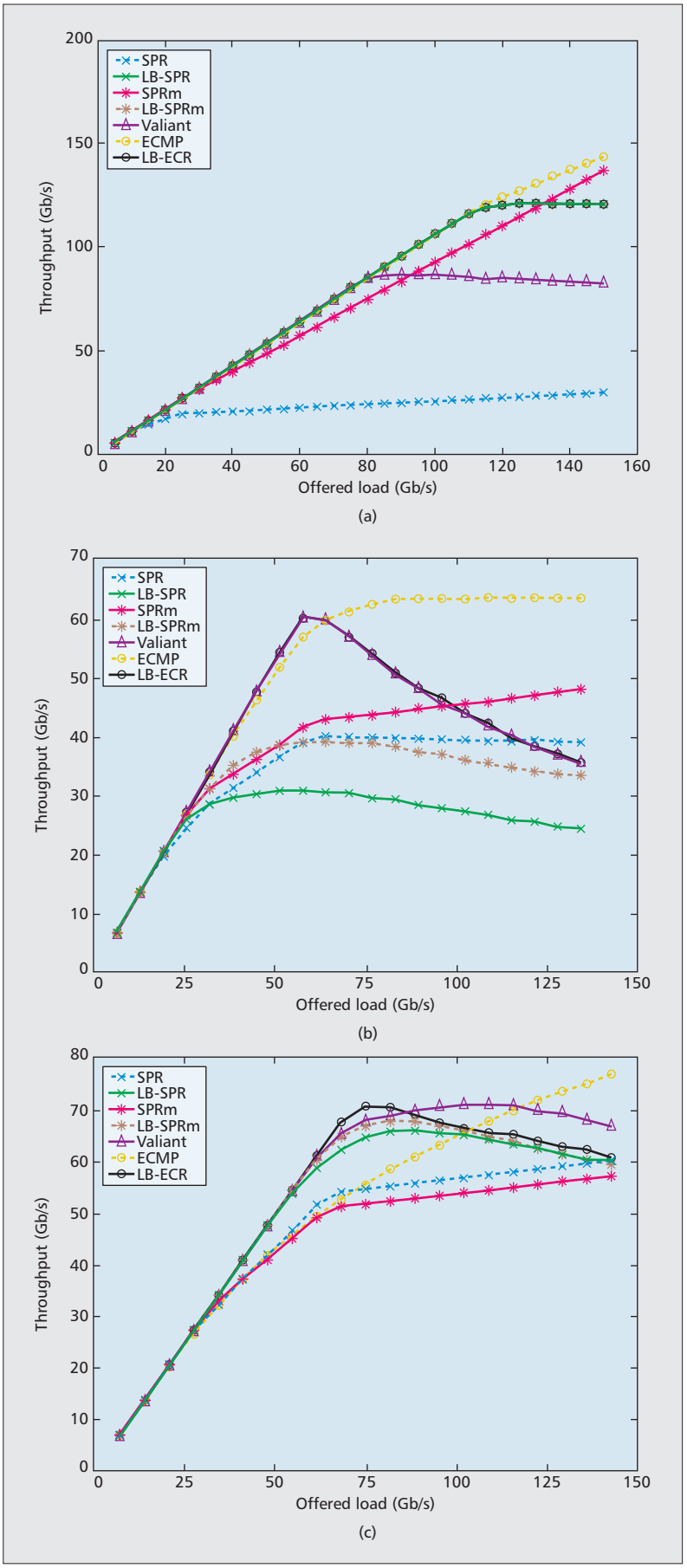

Figure 4. *Throughput: a) performance for the fat-tree topology; b) performance for the flattened butterfly topology; c) performance for the dragonfly topology.*

It has been shown that the optimization of load balancing improves guaranteed throughput in a loss-free network. In particular, the proposed LB-ECR algorithm achieves the highest guaranteed throughputs in loss-free networks.

References

[1] M. Antić and A. Smiljanić, "Routing with Load Balancing: Increasing the Guaranteed Node Traffics," *IEEE Commun. Letters*, June 2009.
[2] M. Antić, N. Maksić, and A. Smiljanić, "Two Phase Load Balanced Routing Using OSPF," *IEEE JSAC*, Jan. 2010.
[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proc. SIGCOMM*, 2008.
[4] C. Guo *et al.*, "Bcube: A High Performance, Server-Centric Network Architecture for Modular Data Centers," *Proc. SIGCOMM*, 2009.
[5] J. Kim *et al.*, "Topology-Driven, Highly- Scalable Dragonfly Topology," *Proc. Int'l. Symp. Computer Architecture 2008*.
[6] J. Kim *et al.*, "Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks," *Proc. ISCA*, 2007.
[7] A. Greenberg *et al.*, "VL2: A Scalable and Flexible Data Center Network," *Proc. SIGCOMM*, 2009.
[8] I. Keslassy *et al.*, "Optimal Load-Balancing," *Proc. INFOCOM*, Mar. 2005.
[9] A. Smiljani c, "Rate and Delay Guarantees Provided by Clos Packet Switches with Load Balancing," *IEEE Trans. Net.*, Feb. 2008.
[10] M. Al-Fares *et al.*, "Hedera: Dynamic Flow Scheduling for Data Center Networks," *Proc. Usenix NSDI 2010*, 2010.
[11] C. Raiciu *et al.*, "Improving Data Center Performance and Robustness with Multipath TCP," *Proc. Usenix NSDI 2011*, 2011.
[12] J. T. Moy, *OSPF Complete Implementation*, Addison-Wesley Professional, 2000.
[13] L. Valiant and G. Brebner, "Universal Schemes for Parallel Communication," *Proc. 13th Annual Symp. Theory of Computing*, May 1981, pp. 263–77.
[14] N. Maksi c, *et al.*, "On the Performance of the Load Balanced Shortest Path Routing," *Proc. PacRim'09*, 2009.
[15] B. Towles and W. J. Dally, "Worst-Case Traffic for Oblivious Routing Functions," *Computer Architecture Letters*, 2002.

Biographies

Nataša Maksić (maksicnatasa@gmail.com) received her B.Sc. degree in electrical engineering from Belgrade University, Serbia, 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. She was awarded a scholarship by the Serbian Ministry of Education from 2003 to 2005 for her academic record, and received several awards from companies such as YUBC Systems and Eurobank EFG.

Aleksandra Smiljanić [M'96] received M.A. and Ph.D. degrees in electrical engineering from Princeton University in 1996 and 1999, respectively. She got her B.Sc. degree in electrical engineering at Belgrade University in 1993. Her area of research is high-performance switching and routing. Currently, she works as an associate professor at Belgrade University. She worked for AT&T Labs Research from 1999 until 2004. She is the author of numerous conference and journal papers in the area of high-performance switching and routing. She is the inventor of 10 U.S. patents in this area. She 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. She served as an Editor of *IEEE Communication Letters* and *OSA Journal on Optical Networking*. She has served as a Guest Editor of the *IEEE JSAC* Special Issue on Switching and Routing for Scalable and Energy Efficient Data Center Networks.