# Load-Balanced Routing via Randomization

*Sangman Bak*
Dept. of Computer Science
University of Houston
Houston, TX 77204-3475, USA
smbak@cs.uh.edu

*Jorge A. Cobb*
Dept. of Computer Science
University of Texas at Dallas
Dallas, TX 75083-0688, USA
jcobb@utdallas.edu

*Ernst L. Leiss*
Dept. of Computer Science
University of Houston
Houston, TX 77204-3475, USA
coscel@cs.uh.edu

## Abstract

Future computer networks are expected to carry bursty traffic. Shortest-path routing protocols have the disadvantage of causing bottlenecks due to their single-path routing approach. That is, the selected shortest path between a source and a destination may become highly congested even when many other possible paths have low utilization. We propose a family of routing schemes that randomly balance traffic over the whole network; in this way, they remove bottlenecks and consequently improve network performance. For each data message to be sent from source s to a destination d, each of the proposed routing protocols randomly chooses an intermediate node e from a selected set of network nodes, and routes the data message along the shortest path from s to e. Then, it routes the data message via the shortest path from e to d. Intuitively, we would expect that this increases the effective bandwidth between each pair of nodes. Our simulation results indicate that the proposed family of load-balanced routing protocols indeed distribute data traffic evenly over the whole network and improve network performance with respect to throughput, message loss, message delay and link utilization.

## 1. Introduction

In a wide-area store-and-forward computer network, such as the Internet, routing protocols are essential. They are mechanisms for finding an efficient path between any pair of source and destination nodes in the network and for routing data messages along this path. The path must be chosen so that network throughput is maximized and message delay and message loss are reduced as much as possible.

There are mainly two types of routing protocols: *source routing* and *destination routing*. In source routing, a source node determines the path that a data message must take [9]. In destination routing, each node uses its routing table to store a preferred neighbor to each destination. Thus, the routing table specifies only one hop along the path from the current node to the destination. In a stable state of the protocols, the path consisting of consecutive preferred neighbors for a given destination is assumed to be the shortest path to the destination.

Destination routing protocols are classified into two types of routing protocols: *distance-vector routing* [15], for example, used in the RIP Internet protocol [11], and *link-state routing* [12], for example, used in the OSPF Internet protocol [13].

Unfortunately, destination routing protocols suffer performance degradation because all data messages are routed via the same shortest path to the destination until the routing tables have been updated. The problem

with these routing protocols is that there are no good mechanisms to alter the routing other than updating the routing tables. They use the same shortest path from the source to the destination to route data messages. This shortest path may be highly congested, even when many other paths to the destination have low link utilization. This congestion may trigger the loss of valuable data messages [19]. Moreover, using the same path to the destination limits the maximum possible throughput between the source and the destination to be at most the minimum capacity of the links along the shortest path from the source to the destination.

A result in network flow theory, known as the max-flow min-cut theorem [4], shows that distributing the traffic load over all available paths between a source and a destination in the network, instead of using only one path of minimum cost, increases the effective bandwidth up to the capacity of the minimum cut separating these two nodes.

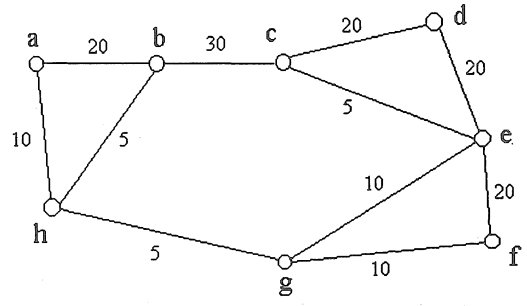


**Figure 1.** A network topology

For example, let's consider Figure 1. The number by each link represents its capacity. Suppose that node **a** wants to send data messages to node **f**. Suppose that we use the hop count in order to calculate the cost of a path in the network. Then the effective bandwidth between node **a** and node **f** is 30, while the effective bandwidth of the unique shortest path (**a-h-g-f**) from node **a** to node **f** is 5.

Several multiple-path routing techniques to increase the effective bandwidth between each pair of nodes and to attempt thereby to improve performance have been proposed ([2], [5], [16], [18], [19]). These routing protocols improve performance by routing data messages via multiple paths to the destination. They provide alternate paths to distribute data traffic when the selected shortest path to the destination becomes congested. Some techniques are Shortest Path First with Emergency Exits [18] based on link-state routing, Multiple Disjoint Paths [16] based on distance-vector routing, and Dynamic Multi-path Routing [2] based on source routing. The disadvantages of these techniques are that they require considerable processing overhead, need significant storage space, or increase the complexity of the routing algorithms. Several randomized multiple-path routing schemes ([6], [14]) have been proposed for regular network topologies, such as mesh, torus, and butterfly, but these schemes are not suitable for the Internet, which has an irregular network topology.

In this paper we propose a simple and efficient routing scheme that improves network performance by distributing the traffic load over all available paths to the destination in an irregular network topology.

The rest of this paper is organized as follows. Section 2 sketches the load-balanced routing protocol. In Sections 3 and 4 we present the simulation model and our results. In Sections 5, we draw conclusions.

# 2. Overview of the Load-Balanced Routing

In this section, we sketch how our method, called Load-Balanced Routing (LBR), routes data messages to the destination. Each node creates data messages and receives data messages from its neighbors. The node should forward these data messages to its neighbors so that the number of links traversed by each data message is as small as possible, while at the same time attempting to distribute these data messages evenly throughout the network to avoid congestion and increase network throughput. Our scheme is based on the distance-vector routing algorithm.

## 2.1 Load-Balanced Routing via Full Randomization

In this subsection, we sketch the Load-Balanced Routing via Full Randomization (LBR-FR) with ideas borrowed from Valiant's randomized routing algorithm [16]. Valiant's scheme was originally developed for a regular network topology such as an n-dimension binary cube of parallel computers, which is different from the Internet topology. The Internet has an irregular topology. Our paper addresses these particular issues in the context of the IP network.

Here is **the algorithm of the Load-Balanced Routing via Full Randomization:**

1. *For each data message to be sent from a source node s to a destination node d, LBR-FR randomly chooses an intermediate node e among all the network nodes.*
2. *It routes the message via the shortest-distance path from s to e.*
3. *It routes the message via the shortest-distance path from e to d.*

As an example, consider Figure 1 again. Suppose that node **a** (source) wants to send data messages to node **f** (destination). For load balancing, node **a** should distribute the data messages uniformly over all possible paths to node **f**. Node **a** may accomplish this by selecting at random an intermediate node (say node **c**) among all the nodes in the network whenever node **a** sends a data message to node **f**, routing it to the intermediate node **c** via the shortest path between node **a** and node **c** and then routing it to destination **f** via the shortest path between node **c** and node **f**.

To accomplish this, each message must carry at least three pieces of information: the destination d, the intermediate node e, and a bit b. Bit b indicates whether the message has not yet reached e (b = 0) or has already passed through node e (b = 1).

Therefore, the operation of the protocol is as follows. Initially, the source node s sends the message with b = 0, and routes it in the direction of node e. As long as b = 0, the message keeps being routed along the

network until it reaches node e. At node e, b is updated to 1, and the message is routed towards node d. As long as b = 1, the message keeps being routed along the network until it reaches node d, where it is delivered.

In this simple version, since the set of candidates for an intermediate node are all the nodes in the network, our routing protocol has full randomization of choosing an intermediate node from all the network nodes. From now on, we call the simple version of LBR Load-Balanced Routing via Full Randomization (LBR-FR).

This technique distributes the traffic load over all the paths between a source and a destination in the network and increases the effective bandwidth up to the capacity of the minimum cut separating these two nodes, which is the upper bound on the available bandwidth between these two nodes [4].

## 2.2 Load-Balanced Routing via Bounded Randomization

The protocol described in Subsection 2.1, however, has a shortcoming for a pair of nodes of short distance. It is possible that a data message is routed to the destination via a very long path, much longer than a shortest path from the source to the destination.

For example, in Figure 1, suppose that node **a** wants to send a data message to node **b** and it randomly chooses node **f** as the intermediate node. As a result, the algorithm routes the data message to node **f** via the shortest path (**a-h-g-f**) and then routes it to node **b** via the shortest path (**f-e-c-b**). Although there is a path of length 1 between node **a** and node **b**, the proposed algorithm results in the use of a path of length 6.

Clearly, routing paths that are excessively long will waste network resources.

To remedy this problem, we introduce a parameter $k$, in order to exclude nodes from being candidates for an intermediate node that are "too far away" from the source. The set of candidates is restricted to all the nodes whose distance from the source is at most $k$.

The value chosen for $k$ affects delay, path length, load balancing, and network throughput. If $k$ is zero, the length of the path is minimized because our routing protocol becomes the conventional distance-vector routing protocol, and thus the data message will be routed via a shortest path to the destination. On the other hand, if $k$ is non-zero, a larger number of routing paths may be available, which alleviates congestion and increases the effective bandwidth between these two nodes, but at the expense of possibly increasing the length of the traveled path. If $k$ is INFINITY, the proposed algorithm is LBR-FR (see Section 2.1). This may increase the effective bandwidth up to the capacity of the minimum cut separating these two nodes.

Choosing an appropriate value for $k$ is crucial for the performance of the algorithm. Choosing too small a value may exclude nodes that are too far away from the source from being candidates for an intermediate node, but it will increase the likelihood of a bottleneck. On the other hand, choosing too large a value may waste network resources by routing packets via excessively long paths, but it will increase the effective bandwidth up to the capacity of the minimum cut separating each pair of nodes. To reach a compromise between these two extremes, the parameter $k$ may be chosen to be the average of the distance to each node reachable from the source (LBR-BR1) [3]:

$$\bullet\ k = \frac{1}{n}\sum_{i=1}^{n} dist(s, d_i)$$

where $d_i$ is a node in the network and s is the source node.

This value is a constant for the source *s*, since each link is considered to have a cost of 1. This value, however, has shortcomings. It limits the effective bandwidth between each pair of the nodes in the network to less than the capacity of the minimum cut separating the pair. The static value of *k* is too strong a restriction for a pair of nodes with a long path length and too weak a restriction for a pair of nodes with a short path length.

To remedy this problem of the static value for the parameter *k*, we may choose the value of the parameter *k* more intelligent to be fair to all the pairs of network nodes and consider a dynamic choice of parameter k (LBR-BR2):

$$\bullet\ k = dist(s,d) * \frac{MAX(dist(s,d_i)) - 1}{MAX(dist(s,d_i))}$$

where *di* is a node in the network, *s* is the source node and *d* is the destination node.

This value of the parameter *k* dynamically changes according to the length of the shortest path from the source node *s* to the destination node *d*.

## 3. Simulation Model

Our simulation studies were done on the Maryland Routing Simulator (MaRS) [1], which is a network simulator developed at the University of Maryland. A network configuration consists of a physical network, a routing algorithm, and a workload.

The routing algorithms are DVR, LSR, LBR-FR, LBR-BR1, and LBR-BR2. DVR is a distance-vector and loop-free routing protocol [15], which uses a shortest-distance path for each pair of source and destination nodes. LSR is a link-state routing protocol [12], where each node calculates and broadcasts the costs of its outgoing links periodically and Dijkstra's shortest path algorithm [8] is applied to the view of the network topology to determine next hops. To get a better understanding of all LBR protocols, we compare the performance of the three LBR protocols formulated in Subsection 2.2 against DVR and LSR.

In our simulation, the assumed physical network is the NSFNET topology given in Figure 2 [7]. All links have a bandwidth of 1.5 Mbits/sec. We assumed that there are no link or node failures. Each node has a buffer space of 50,000 bytes. The processing time of a data message at each node equals 1 μsec. In order to calculate the cost of a path in the network, we use the hop count and the link utilization. When we use the hop count as a link cost, the cost of each link is 1. The propagation delay of each link is 1 msec.

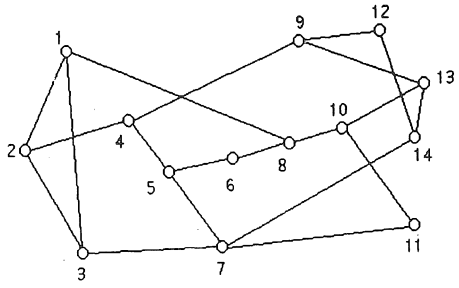


**Figure 2.** NSFNET Topology: 14 nodes, 21 bi-directional links, average degree of 3

The workload is the file transfer protocol (FTP). All FTP connections have the following parameters: the data message length equals 512 bytes, the inter-message generation time is 1 or 10 msec, and the window size is 500 messages. Connections start when the simulation begins and they are considered never-ending.

We consider performance measures of throughput, message delay, message loss and link utilization. The measurement interval of each simulation is 50,000 msec.

- *Throughput*: the total number of data bytes acknowledged during the measurement interval divided by the length of the measurement interval.
- *Average message delay*: the total delay of all data messages acknowledged during the measurement interval divided by the number of data messages acknowledged during the measurement interval.
- *Instantaneous message delay*: the total delay of all data messages acknowledged in the last update period divided by the number of data messages acknowledged in this period.
- *Message loss*: the total number of the messages dropped during the measurement interval.
- *Link utilization*: the data service rate divided by the link bandwidth.

# 4. Simulation Results

Figure 3 shows throughput versus the number of connections. The throughput in all the routing protocols in general increases as the number of connections increases. With respect to throughput, the three LBR protocols are much better than DVR and LSR, both when the number of connections is low and high. DVR and LSR perform about the same except when the number of connections is 20. The throughput generally increases linearly except around the saturation points. The system is saturated when the number of connections is around 2 and 10 in LBR protocols, while the system is saturated when the number of connections is around 2, 11 and 15 in DVR and LSR.

Figure 4 shows the average message delay versus the number of connections. DVR and LSR exhibit higher delay oscillations than our LBR protocols. DVR and LSR first increase sharply and level off as the system becomes congested. The three LBR protocols always have lower average delay than DVR and LSR during the measurement interval except when the number of connections is 20. However, in both Figure 3 and Figure 4, the curves tend to converge as the number of connections increases. This is logical, since even a

single path routing algorithm would tend to balance traffic load over the entire network (from a global perspective), as the number of connections approaches its maximum N·(N-1), where N is the number of the network nodes.

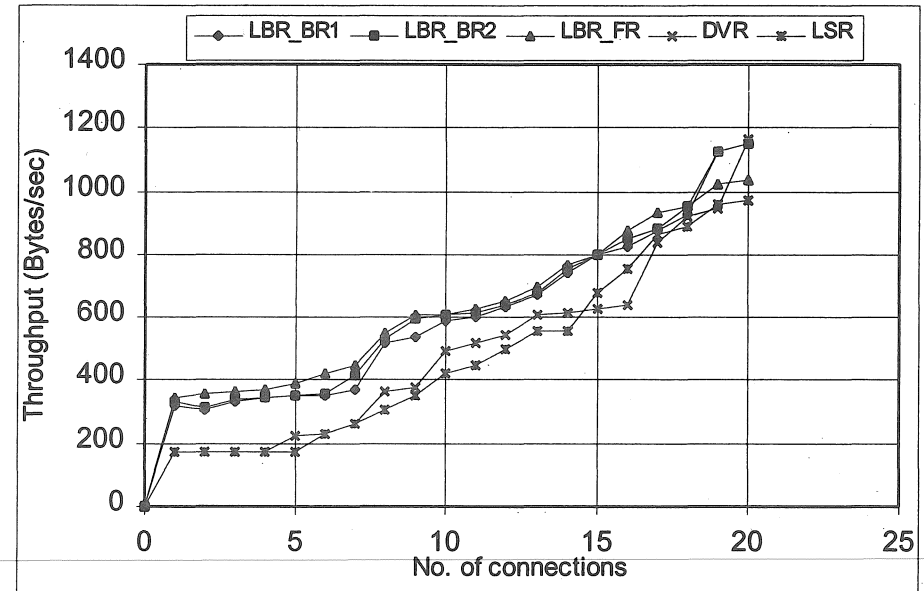


Figure 3: Throughput vs. No. of connections

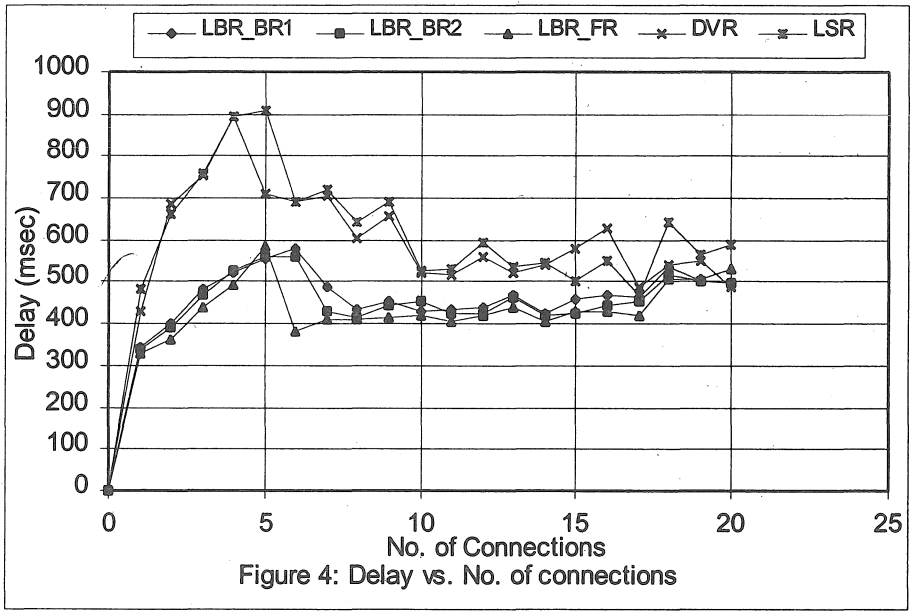


Figure 4: Delay vs. No. of connections

Figure 5 shows the message loss versus the number of connections. The message loss in all the LBR protocols is lower than in DVR and LSR both when the numbers of connections are low and high (except for 20).

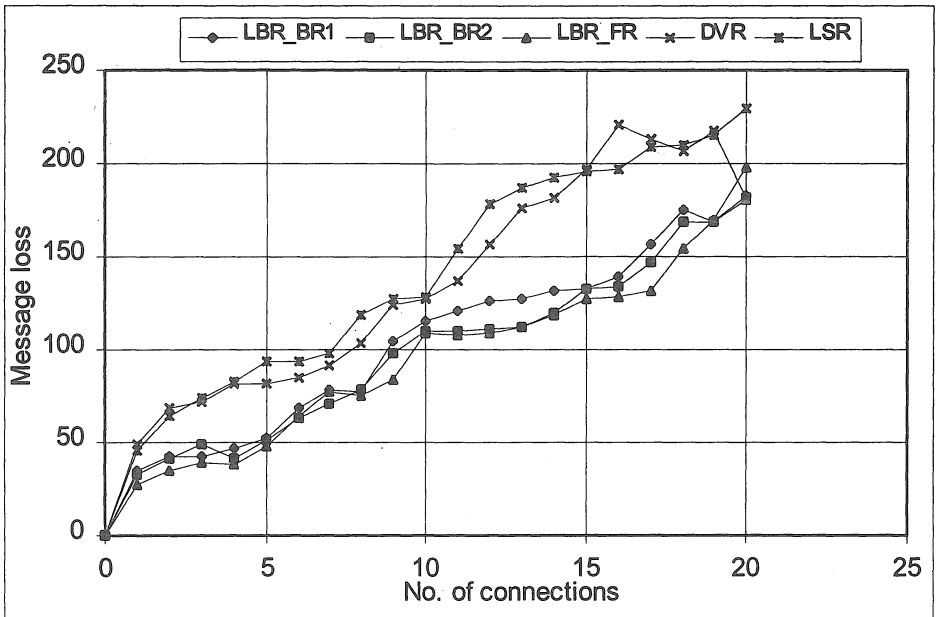


Figure 5: Message loss vs. No. of connections

Figure 6 shows instantaneous message delay versus time. The instantaneous message delay for DVR and LSR is always higher than that for the family of LBR protocols during the measurement interval. The instantaneous message delay for DVR and LSR exhibits bigger oscillations.

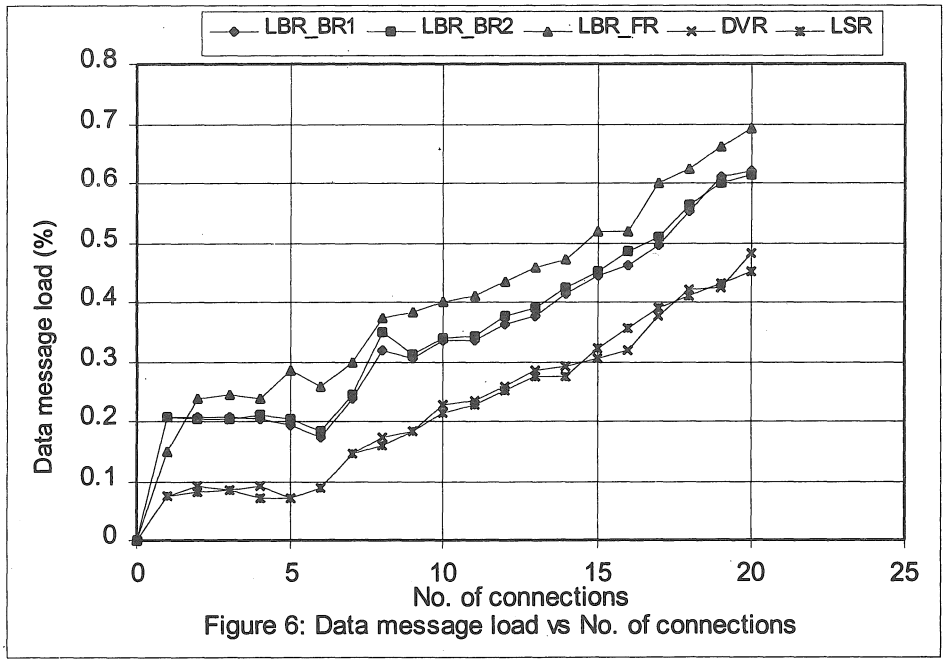


Figure 6: Data message load vs No. of connections

Figure 7 shows throughput versus number of connections in the hot spot. In this scenario, we make a special node, called hotspot, have many more connections than any other nodes in the network. All the LBR protocols have better throughput than DVR and LSR do until the number of connections reaches 10. LBR-FR

has the highest throughput when the number of connections is in the range from 1 to 9, while LBR-BR1 has the highest throughput when the number of connections is more than 9.

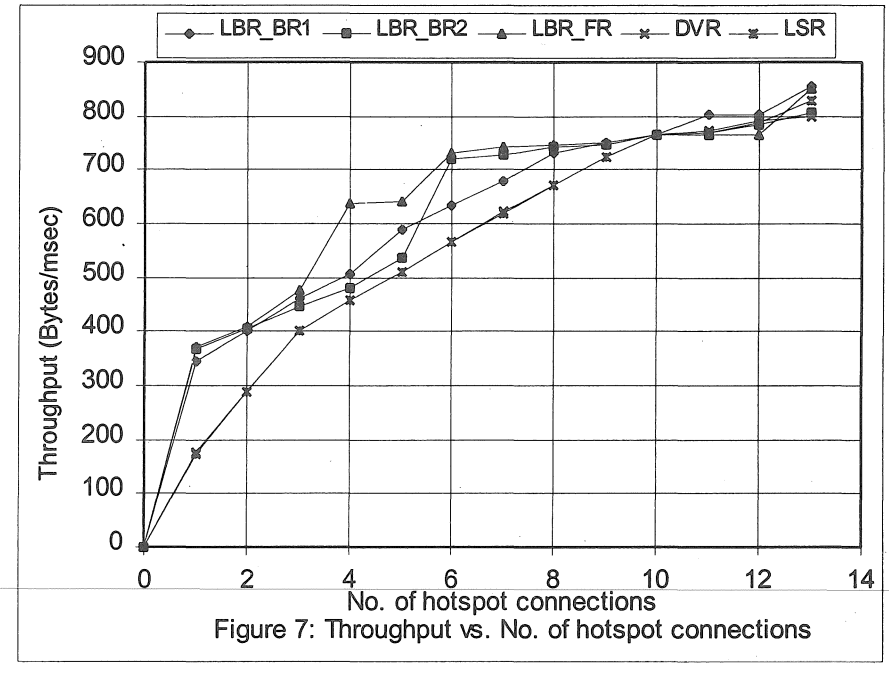


Figure 7: Throughput vs. No. of hotspot connections

Figure 8 shows throughput versus time when the number of connections is 7. DVR and LSR have about the same network throughput during the measurement interval. DVR and LSR have lower throughput than the LBR protocols at all times during the measurement time.

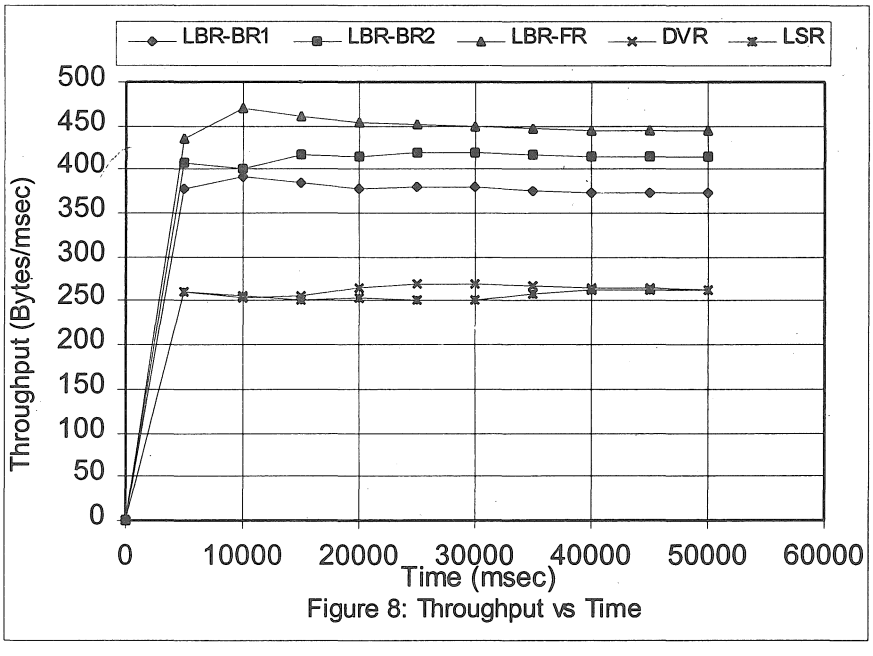


Figure 8: Throughput vs Time

Table 1 shows the average link utilization during the measurement interval when the number of connections is 7. We can see that the LBR protocols distribute the data messages more uniformly over the

whole network than DVR and LSR by inspecting the link utilization over the whole network. Also, the total and the average of the link utilization indicate that the LBR protocols use network resources more productively than DVR and LSR (see Figure 3).

| Link | Link Utilization | | | | |
|---|---|---|---|---|---|
| | LBR-BR1 | LBR-BR2 | LBR-FR | DVR | LSR |
| (1,2) | 0.077892 | 0.066935 | 0.063761 | 0 | 0 |
| (2,1) | 0.145135 | 0.075503 | 0.148753 | 0 | 0 |
| (1,3) | 0.000341 | 0.000341 | 0.000683 | 0 | 0 |
| (3,1) | 0.183364 | 0.157832 | 0.280271 | 0 | 0 |
| (1,8) | 0.250675 | 0.163226 | 0.430012 | 0 | 0 |
| (8,1) | 0.00041 | 0.000649 | 0.070656 | 0 | 0 |
| (2,3) | 0.138342 | 0.222072 | 0.274193 | 0.012288 | 0.0121173 |
| (3,2) | 0.770499 | 0.798265 | 0.774193 | 0.873574 | 0.9145045 |
| (2,4) | 0.992376 | 0.995225 | 0.993679 | 0.992854 | 0.994391 |
| (4,2) | 0.254259 | 0.167083 | 0.260779 | 0.026931 | 0.0241664 |
| (3,7) | 0.296107 | 0.296245 | 0.536413 | 0 | 0 |
| (7,3) | 0.054306 | 0.045363 | 0.115268 | 0 | 0 |
| (4,5) | 0.39994 | 0.344952 | 0.434227 | 0.165342 | 0.1093632 |
| (5,4) | 0.65956 | 0.894758 | 0.705982 | 0.83151 | 0.8682832 |
| (4,9) | 0.983099 | 0.994362 | 0.984521 | 0.993485 | 0.9936437 |
| (9,4) | 0.142575 | 0.16186 | 0.229717 | 0.060928 | 0.0303445 |
| (5,6) | 0.253235 | 0.28986 | 0.30406 | 0 | 0.0017408 |
| (6,5) | 0.141688 | 0.085197 | 0.162065 | 0.042257 | 0.0422912 |
| (5,7) | 0.238831 | 0.037615 | 0.259686 | 0.042257 | 0.0422911 |
| (7,5) | 0.141687 | 0.19398 | 0.233028 | 0 | 0.0323925 |
| (6,8) | 0.127044 | 0.210193 | 0.218044 | 0 | 0 |
| (8,6) | 0.355704 | 0.346351 | 0.416836 | 0.383045 | 0.3813377 |
| (7,11) | 0.170462 | 0.102093 | 0.204322 | 0.042257 | 0.0422911 |
| (11,7) | 0.198895 | 0.18118 | 0.214938 | 0 | 0.0017408 |
| (7,14) | 0.406221 | 0.226372 | 0.466125 | 0 | 0 |
| (14,7) | 0.051029 | 0.053965 | 0.043725 | 0 | 0.0306859 |
| (8,10) | 0.239445 | 0.298701 | 0.447933 | 0 | 0 |
| (10,8) | 0.22016 | 0.279825 | 0.315222 | 0.383625 | 0.3819178 |
| (9,12) | 0.435405 | 0.481621 | 0.379802 | 0.451209 | 0.4610737 |
| (12,9) | 0.055535 | 0.052463 | 0.141346 | 0.027921 | 0.0132437 |
| (9,13) | 0.31546 | 0.347751 | 0.32669 | 0.33693 | 0.3300694 |
| (13,9) | 0.348228 | 0.285324 | 0.472956 | 0.020719 | 0.0049835 |
| (10,11) | 0.000819 | 0.001126 | 0.001161 | 0 | 0 |
| (11,10) | 0.356181 | 0.348194 | 0.457319 | 0.383659 | 0.3819519 |
| (10,12) | 0.134622 | 0.15616 | 0.197495 | 0 | 0 |
| (12,10) | 0.043486 | 0.051336 | 0.02263 | 0 | 0 |
| (10,13) | 0.316417 | 0.300135 | 0.423118 | 0 | 0 |
| (13,10) | 0.039322 | 0.047138 | 0.027682 | 0 | 0 |
| (12,14) | 0.000444 | 0.000853 | 0.040585 | 0 | 0.0152917 |
| (14,12) | 0.068471 | 0 | 0.171759 | 0 | 0 |
| (13,14) | 0.000683 | 0.000888 | 0.02048 | 0 | 0.0153941 |
| (14,13) | 0.287266 | 0.172988 | 0.311125 | 0 | 0 |
| Total | 10.29562 | 9.935979 | 12.58324 | 6.070791 | 6.1255107 |
| Average | 0.245134 | 0.236571 | 0.299601 | 0.144543 | 0.1458455 |
| Variance | 0.056822 | 0.066034 | 0.058172 | 0.081622 | 0.0842279 |

Table 1: Average link utilization (No. of connections = 7)

## 5. Conclusions

We presented a family of load-balanced routing protocols to distribute the data traffic randomly over all available paths to a destination in the network for balancing data load. Our simulation results show that the proposed family of routing protocols has improved performance with respect to throughput, message loss, message delay and link utilization at most times during the measurement interval, compared with DVR and LSR, which are conventional destination routing protocols. The proposed family of routing protocols is simple and with little control overhead, since it randomly chooses one node within $k$ hops as an intermediate node and doesn't use the current traffic status of the network. When the traffic is very balanced in the whole network, since our routing method may use longer path than the shortest path, the gain from load balance may be smaller than the pay from longer delay. Also there may be negative effects of the randomized algorithm on applications that require guaranteed performance. For example, the worst-case performance for our randomized scheme may be worse than the theoretical worst case for the plain shortest-path approaches. Addressing these issues within the context of our randomized network routing protocol is the object of future work.

## References


[1] C. Alaettinoglu., K. Dussa-Zieget, I. Matta,, O. Gudmundsson, and A. U. Shankar, *MaRS – Maryland Routing Simulator* Version 1.0. Department of Computer Science, University of Maryland, 1991.

[2] S. Bahk, and M. E. Zarki, *Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks*, Proceedings of the 1992 ACM SIGCOMM Conference, Vol. 22, Oct. 1992.

[3] S. Bak and J. A. Cobb, *Randomized Distance-Vector Routing Protocol*, Proceedings of ACM Symposium on Applied Computing, San Antonio, Texas, Feb. 1999.

[4] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.

[5] J. A. Cobb and Gouda M. G., *Balanced Routing*, IEEE Proceedings of the International Conference on Network Protocols, 1997.

[6] R. Cole, B.M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A.W. Richa, K. Schroeder, R.K. Sitaraman, and B. Voecking, *Randomized Protocols for low-congestion circuit routing in multistage interconnection networks*, Procceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 378-388, May 1998.

[7] D. E. Comer, Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture, Prentice Hall, 1995.

[8] E. W. Dijkstra, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik, Vol. 1, pp. 269- 271, 1959.

[9] R.C. Dixon and D.A. Pitt, *Addressing, Bridging, and Source Routing (LAN interconnection)*, IEEE Network, Vol. 2, No. 1, Jan.1988.

[10] M. Gouda, *The Elements of Network Protocol Design*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., 1998.

[11] G. Malkin, RIP Version 2, *Internet Request for Comments 1723,* Nov. 1994, Available from http://www.ietf.cnri.reston.va.us.

[12] J.M. McQuillan, Ira Richer and E.C. Rosen, *The New Routing Algorithm for the ARPANET*, IEEE Trans. on Communications, Vol. COM-28, N0. 5, pp. 711-719, May 1980.

[13] J. Moy, *Ospf Version 2*, Internet Request for Comments *1583*, Mar. 1994. Available from http://www.ietf.cnri.reston.va.us.

[14] T. Nesson and S. L. Johnsson, *ROMM Routing on Mesh and Torus Networks*, Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures, July 1995.

[15] Segall and M. Sidi, *A Failsafe Distributed Protocol for Minimum Delay Routing*, IEEE Trans. on Commun., COM-29(5), 686-695, May 1981.

[16] D. Sidhu, R. Nair, S. Abdallah, *Finding Disjoint Paths in Networks*, Proceedings of the 1991 ACM SIGCOMM Conference, 1991.

[17] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.

[18] Z. Wang, J. Crowcroft, *Shortest Path First with Emergency Exists*, Proceedings of the 1990 ACM SIGCOMM Conference, 1990.

[19] W. T. Zaumen and J. J. Garcia-Luna-Aceves, *Loop-Free Multipath Routing Using Generalized Diffusing Computations*, Proc. IEEE INFOCOM 98, San Francisco, California, Mar. 29, 1998.