Load-Balanced Routing via Bounded Randomization

Sangman Bak
Dept. of Computer Science
University of Houston
Houston, TX 77204-3475
713-743-3350
smbak@cs.uh.edu

Jorge A. Cobb
Dept. of Computer Science
University of Texas at Dallas
Dallas, TX 75083-0688
972-883-2479
jcobb@utdallas.edu

Ernst L. Leiss
Dept. of Computer Science
University of Houston
Houston, TX 77204-3475
713-743-3350
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. That is, the shortest path between a source and a destination may become highly congested even when many other paths have low utilization. We propose a routing scheme that distributes traffic over the whole network via bounded randomization; therefore, it removes bottlenecks and increases network throughput. For each data message to be sent from a source s to a destination d, the proposed routing protocol 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 this load-balanced routing protocol distributes traffic evenly over the whole network and, in consequence, increases network throughput.

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 [8]. 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* [14], for example, used in the RIP Internet protocol [10], and *link-state routing* [11], for example, used in the OSPF Internet protocol [12].

Destination routing protocols suffer performance degradation because all data messages are routed via the same shortest path to the destination as long as the routing tables are unchanged. The problem with these routing protocols is that there are no mechanisms for altering the routing other than updating the routing tables. The 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 due to the buffer overflow at each node. Using a single path to the destination limits the maximum throughput possible 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.

Maximizing network throughput is an important goal in the design of routing protocols. If the network uses shortest routing protocols to carry bursty traffic, then many of these data packets would be dropped due to the limited buffer space of each node when these shortest paths are congested. In this paper, we want to minimize the packet loss due to the buffer overflow at each node. We also want to maximize the network throughput. Our approach increases the effective bandwidth between the source and the destination so that more data packets can be delivered. A result in network flow theory, known as the max-flow min-cut theorem [5], 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.

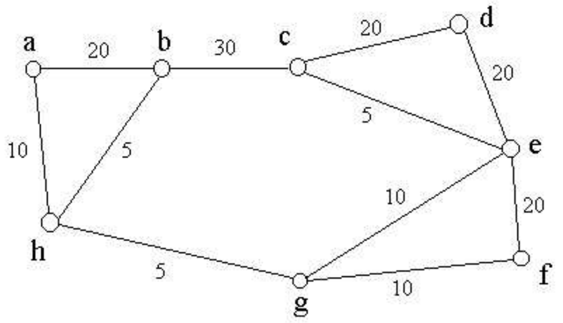

Figure 1. Network topology

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 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], [6], [15], [17], [18]). 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 [17] based on link-state routing, Multiple Disjoint Paths [15] 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 ([4], [13]) 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 a recent paper [3], we proposed a randomized routing scheme that improves network performance by randomly distributing the traffic load over a set of paths to the destination. The routing protocol was formulated for an IP (Internet Protocol) network with an irregular topology.

We propose an improved method that implements our routing approach in order to maximize the number of packets that reach their destination and to minimize the number of packets that are dropped due to buffer overflow at each network node.

The rest of this paper is organized as follows. Section 2 sketches the load-balanced routing protocol. Section 3 introduces the protocol notation to give a formal version of our routing protocol, which is given in Section 4. In Sections 5 and 6, we present the simulation model and our results, in Sections 7 and 8, we outline future work and draw conclusions, respectively.

2. Overview of the Load-Balanced Routing

In this section, we sketch how our method, called the 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 (the cost of a path) 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 packet 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 packet via the shortest-distance (or least-cost) path from s to e.*
3. *It routes the packet via the shortest-distance (or least-cost) 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 node 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.

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 [5].

2.2 Load-Balanced Routing via Bounded Randomization

The protocol described in Subsection 2.1 has a shortcoming for pairs 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, LBR-FR 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**, LBR-FR 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 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]:

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

  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):

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

where d_i 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. Protocol Notation

In this paper, we use a simple notation to define our routing protocol. A protocol is defined by a set of processes, p[0], p[1], . . . , p[n-1]. A process corresponds to a node in a computer network. A pair of neighboring processes is joined by a communications channel. Henceforth, we use the term process and node interchangeably.

A process is defined by a set of constants, a set of inputs, a set of variables and a set of actions. The actions of a process are separated by the symbol [], as follows:

begin *action.1* [] *action.2* [] . . . [] *action.m* **end**

An action has the following form: *guard* $\rightarrow$ *statement*. A guard is a Boolean expression, which refers to constants, inputs, and variables of the process. A statement is defined recursively as one of the following: skip, assignment statement, conditional (**if** ... **fi**), bounded loop (**for** ... **rof**) and a sequence of two or more statements separated by ";".

An action in a process is enabled if and only if the action's guard is true at the current state of the network. An execution step of a protocol consists of choosing any enabled action from any process, and executing the action's statement. Executions are maximal, i.e., either they consist of an infinite number of execution steps, or they terminate in a state in which no action is enabled. Executions are assumed to be fair, i.e., each action that remains continuously enabled is eventually executed.

The communication between processes is based on a message-passing model. For every pair of neighboring processes p[i] and p[j], there are a FIFO channel from p[i] to p[j] and a FIFO channel from p[j] to p[i]. The statement **send** data(var) **to** p[j] in process p[i] appends a message of type data to the channel from p[i] to p[j], and the field in the message is the current value of variable var in process p[i].

In addition to Boolean expressions, guards in each process p[i] are allowed to be of the form **rcv** data(var) **from any** p[j]. This guard is enabled iff there is a message of type data at the head of an incoming channel of p[i]. If an action with this receive guard is chosen for execution, then, before its command is executed, the data message is removed from the channel, and its field is copied into the local variable var. Furthermore, variable j is set to the identity of the neighbor from whom the message is received.

Similar protocol notations are defined in [6] and [9].

4. Specification of the Load-Balanced Routing Protocol

In this section, we present a formal version only for LBR-BR2. Formal versions of LBR-FR and LBR-BR1 [3] can be derived in a straightforward manner from that of LBR-BR2. Each process has a constant n with the number of the processes in the network and an input set N with the identities of its neighbors.

Each process p[i] has several variables. The variable inter stores candidates for intermediate nodes. That is, inter stores the process id’s of processes which are at most k hops away from process p[i]. Variable rtb[j] stores the preferred neighbor to reach destination p[j], and hop[j] stores the distance to reach destination p[j].

The load-balanced routing protocol (LBR-BR2) is defined as follows.

```
process p [i: 0 . . n-1]
constants
n       :       integer                                {number of nodes in the network}
inputs
N       :       set of {j | p[j] is a neighbor of p[i]}
variables
k       :       0 . . n-1,                             {maximum length of 1st routing path}
inter   :       set of {0 . . n-1}                     {possible intermediate nodes of routing paths}
rtb     :       array [0 . . n-1] of 0 . . n,          {routing table}
hop     :       array [0 . . n-1] of 0 . . n,          {hop count to each destination}
h       :       array [0 . . n-1] of 0 . . n,          {neighbor's hop count to each destination}
e, d    :       0 . . n-1,                             {message's intermediate node and destination}
b       :       0 . . 1,          {status bit: b=0 on 1st routing path, b=1 on 2nd routing path}
x       :       0 . . n-1,
j       :       element of N
begin
  true →
        m := max{hop[x] | 0 ≤ x < n  ∧ 0 ≤ hop[x] < n};
[] true →                                  {create and route a new message to any destination}
        b := 0;
        d := any;
        k := hop[d]*(m-1)/m;                   {determine a value of the parameter k}
        inter := {x | hop[x] ≤ k}
        e := random(inter);
        RTMSG
[] rcv data(b, e, d) from any p[j] → RTMSG
[] true →
        for each j in N do                     {send hop count to neighbors}
                send upd(hop) to p[j]
        rof
[] rcv upd(h) from any p[j] →
        UPDTBL
end
```

Statement RTMSG is defined as follows.

```
if d = i →                                     {arrived, deliver message}
                skip
[] d ≠ i ∧ b = 0 ∧ hop[e] = n →                {unreachable intermediate node}
                skip
```

```
[] $d \neq i \wedge b = 0 \wedge hop[e] < n \wedge e \neq i \rightarrow$          {reachable intermediate node}
                send data(b, e, d) to rtb[e]
[] $d \neq i \wedge b = 0 \wedge hop[e] < n \wedge e = i \rightarrow$          {end of first routing path}
                send data(1, e, d) to rtb[d]
[] $d \neq i \wedge b = 1 \wedge hop[d] = n \rightarrow$                  {unreachable destination}
                skip
[] $d \neq i \wedge b = 1 \wedge hop[d] < n \rightarrow$                  {reachable destination}
                send data(1, e, d) to rtb[d]
fi
```

Statement UPDTBL is defined as follows.

```
hop[i] := 0;
for each x, where $x \neq i$, do
    if $rtb[x] = j \wedge (h[x]+1) \neq hop[x] \rightarrow$     {p[i] currently routes to p[x] via p[j],and
                                                      p[j]'s distance to p[x] has changed}
        hop[x] := min(h[x]+1, n)
    [] $rtb[x] \neq j \wedge (h[x]+1) < hop[x] \rightarrow$           {found a shorter path}
        hop[x] := min(h[x]+1, n);
        rtb[x] := j
    [] $rtb[x] \neq j \wedge (h[x]+1) \geq hop[x] \rightarrow$          {keep the current path}
        skip
    [] $rtb[x] \notin N \rightarrow$                              {p[rtb[x]] is down}
        hop[x] := min(h[x]+1, n);
        rtb[x] := j
    fi;
rof
```

5. 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 [3], and LBR-BR2. DVR is a distance-vector and loop-free routing protocol [14], which uses a shortest-distance path for each pair of source and destination nodes. LSR is a link-state routing protocol [11], where each node calculates and broadcasts the costs of its outgoing links periodically and Dijkstra’s shortest path algorithm [7] is applied to the view of the network topology to determine next hops. To get a better understanding of LBR-BR2 protocol, we compare the performance of the LBR-BR2 protocol against LBR-FR, LBR-BR1, DVR and LSR.

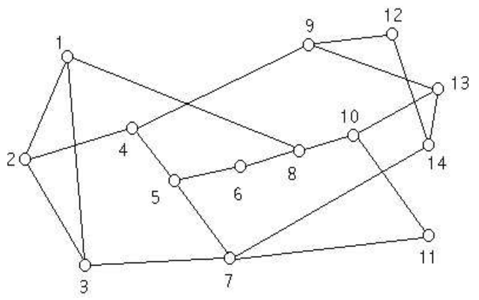

Figure 2. NSFNET Topology: 14 nodes, 21 bi-directional links, average degree of 3

In our simulation, the assumed physical network is the NSFNET topology given in Figure 2. 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. 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.

The workload consists of FTP (file transfer protocol) and telnet connections. A connection is a communication session established between end-user applications at source and destination nodes. All FTP and telnet 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. Traffic was introduced into the network by the FTP and/or telnet connections at the nodes they were attached to. Network traffic consisted of data messages sent from the source of a connection to the destination and response messages sent from the destination of the connection to the source. Further, each source and destination node send acknowledgements for data messages received. Also present in the network are routing messages, which are sent periodically to update the state of the network. Connections start when the simulation begins and they are considered never-ending.

We consider the performance measures of throughput, message delay, message loss, and link utilization. The measurement interval of each simulation is 100,000 msec.

- *Throughput*. The total number of data bytes acknowledged during the measurement interval divided by the length of the measurement interval.
- *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.
- *Message loss*. The total number of messages dropped during the measurement interval.
- *Link utilization.* The data service rate divided by the link bandwidth.

6. Simulation Results

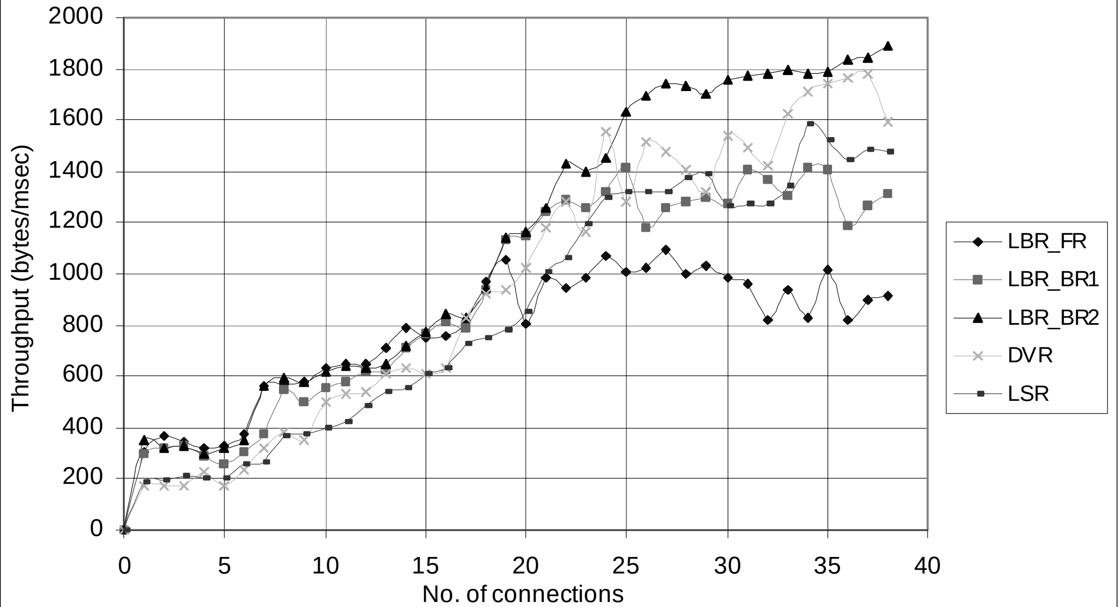

Figure 3. Throughput vs. No. of connections

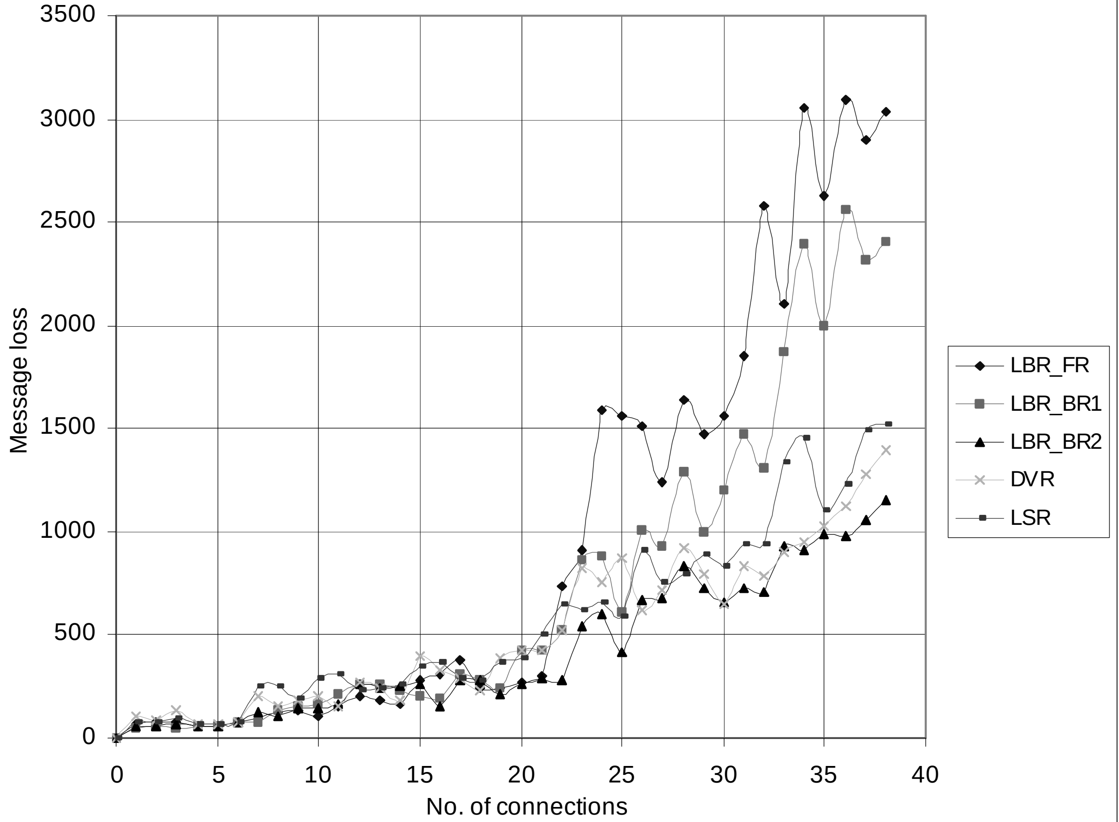

Figure 4. Message loss vs. No. of connections

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, when the number of connections is low. The throughput of the LBR-BR2 is generally highest among all the protocols when the number of connections is in the range from 19 to 38. The throughput increases almost linearly except around the saturation points. The system is saturated when the number of connections is around 2, 10 and 30 in LBR protocols, while the system is saturated when the number of connections is around 2, 11 and 26 in DVR and LSR.

Figure 4 shows the message loss versus the number of connections. The message loss in the LBR-BR2 protocol is generally lower than in the other routing protocols both when the numbers of connections are low and high. LBR-FR has the highest message loss when the numbers of connections are high, while it has a lower message loss when the number of connections is low.

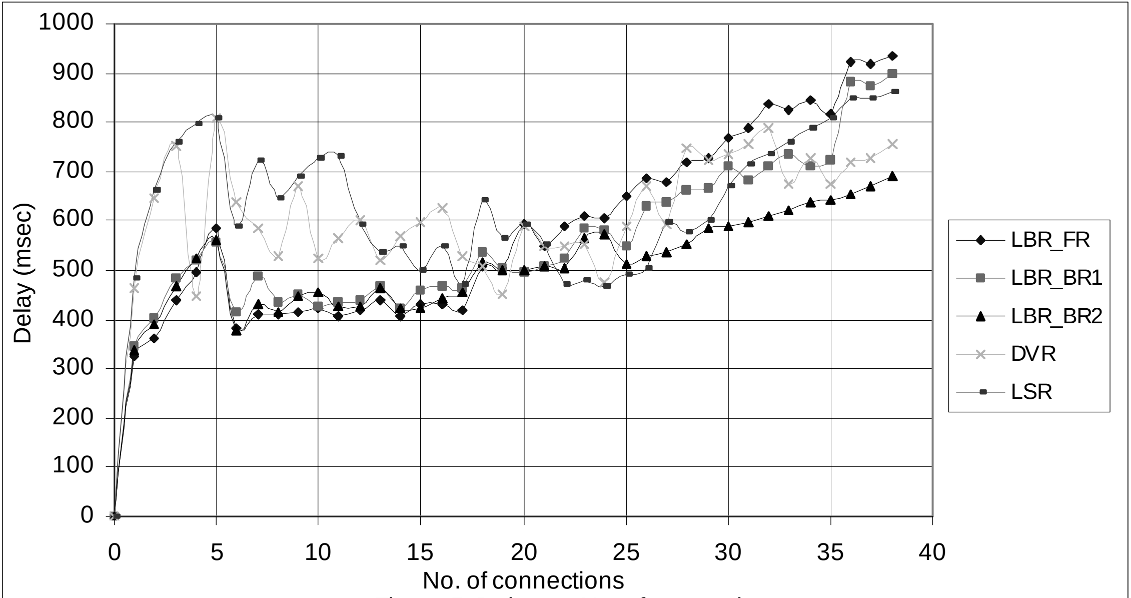

Figure 5. Delay vs. No. of connections

Figure 5 shows the average delay versus the number of connections. DVR and LSR exhibit higher delay oscillations than our LBR protocols. The average delay in all the routing protocols first increases sharply and levels off as the number of connections increases. The three LBR protocols have lower average delay than DVR and LSR during the measurement interval when the number of connections is low. The LBR-BR2 protocol has a low average delay both when the number of connections is high and low. LBR-FR has the highest average delay at most times during the measurement interval when the number of connections is high. However, in both Figures 3 and 5, the curves in LBR-BR2, DVR and LSR tend to converge as the number of connections increases.

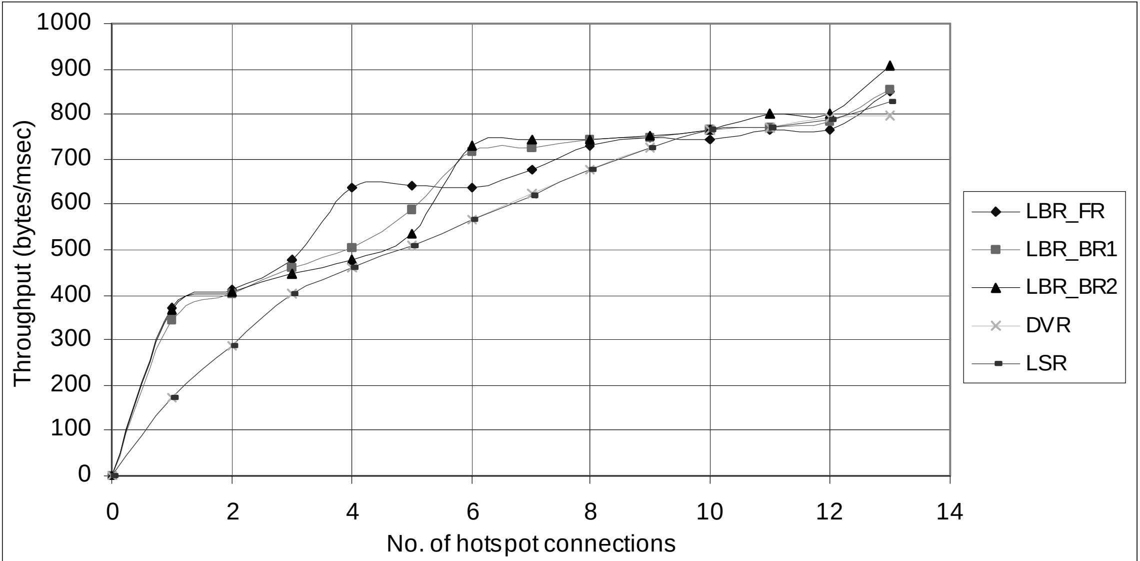

Figure 6. Throughput vs. No. of hotspot connections

Figure 6 shows throughput versus number of connections in the hot spot. In this scenario, we make a special node, called hotspot, have much 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 is 9. LBR_FR has the highest throughput when the number of connections is in the range from 1 to 5, while LBR_BR2 has the highest throughput when the number of connections is more than 5. Table 1 shows the average link utilization during the measurement interval when the number of connections is 8. By inspecting the link utilization over the whole network, we can see that the LBR protocols distribute the data messages more uniformly over the whole network than DVR and LSR. 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).

We shall now discuss our observations and the interesting insights obtained the results.

The performance of LBR-BR1 and LBR-FR is getting worse as the number of connections increases. Especially, LBR-FR exhibits a good performance with respect to throughput, message loss and message delay when the number of connections is low (in the range from 1 to 19), while it exhibits the worst performance among all the routing protocols when the number of connections is high (in the range from 20 to 38). LBR-FR will increase the effective bandwidth up to the capacity of the minimum cut separating these two nodes, while it has a much more chance than LBR-BR1 and LBR-BR2 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. The performance of shortest-path routing protocols (DVR and LSR) is getting better as the number of connections increases. Especially, DVR exhibits bad performance with respect to throughput, message loss, and message delay when the number of connections is low (in the range from 1 to 16), while it exhibits a good performance when the number of connections is high (in the range from 17 to 38).

This is to be expected, since even a single path routing algorithm would tend to distribute load over the entire network (from a global perspective), as the number of connections approaches n•(n-1), where n is the number of the network nodes.

Link	Link Utilization				
	LBR-FR	LBR-BR1	LBR-BR2	DVR	LSR
(1,2)	0.121856	0.220536	0.196233	0	0
(2,1)	0.198315	0.130696	0.152132	0	0
(1,3)	0.002936	0.001161	0.000341	0	0
(3,1)	0.662169	0.425923	0.401546	0	0
(1,8)	0.731325	0.328432	0.356318	0	0
(8,1)	0.004335	0.001058	0.000376	0	0
(2,3)	0.90508	0.901697	0.840716	0.899904	0.904239
(3,2)	0.261973	0.153156	0.193707	0.020139	0.014029
(2,4)	0.995069	0.995623	0.992131	0.992315	0.99268
(4,2)	0.222276	0.179644	0.267503	0.043145	0.029218
(3,7)	0.954249	0.909037	0.95026	0	0
(7,3)	0.266854	0.003891	0.00297	0	0
(4,5)	0.278972	0.239445	0.247228	0.298565	0.147046
(5,4)	0.625362	0.651136	0.658512	0.463428	0.897757
(4,9)	0.99428	0.994313	0.990944	0.990906	0.991348
(9,4)	0.248013	0.162099	0.15005	0.09728	0.032939
(5,6)	0.561359	0.710283	0.656828	0.643448	0.64396
(6,5)	0.28119	0.143053	0.163806	0.067823	0.069427
(5,7)	0.434552	0.298086	0.419294	0.408406	0.0782
(7,5)	0.58737	0.765582	0.821679	0.644643	0.675977
(6,8)	0.18036	0.194901	0.129672	0	0
(8,6)	0.543098	0.271053	0.28003	0	0
(7,11)	0.326042	0.317713	0.374955	0.032017	0.03369
(11,7)	0.324915	0.258867	0.269346	0.305118	0.307439
(7,14)	0.935556	0.802533	0.854488	0.340003	0.008772
(14,7)	0.066389	0.122368	0.112128	0	0.028536
(8,10)	0.556308	0.445645	0.38714	0	0
(10,8)	0.192068	0.195004	0.181555	0	0
(9,12)	0.163772	0.165001	0.207258	0.256785	0.444314
(12,9)	0.229274	0.04096	0.042769	0.052838	0.013005
(9,13)	0.237705	0.2176	0.199202	0.395162	0.312013
(13,9)	0.853144	0.85779	0.86014	0.024303	0.005905
(10,11)	0.001468	0.133666	0.000922	0	0
(11,10)	0.378914	0.042018	0.446805	0	0
(10,12)	0.203844	0.619803	0.129707	0	0
(12,10)	0.051166	0.061713	0.035499	0	0
(10,13)	0.634379	0.001263	0.607034	0	0
(13,10)	0.046012	0.400964	0.049766	0	0
(12,14)	0.002355	0.002185	0.001912	0	0.014916
(14,12)	0.399736	0.22883	0.22982	0.340003	0.004676
(13,14)	0.039629	0.001468	0.001502	0	0.013619
(14,13)	0.510157	0.454349	0.515277	0	0.004096
Total	16.21382	14.05054	14.3795	7.316228	6.667801
Average	0.386043	0.334537	0.342369	0.174196	0.158757
Variance	0.091891	0.093938	0.095212	0.08218	0.093452

Table 1. Average link utilization (No. of connections = 8)

7. Future Work

In previous work [3], we had proposed a randomized distance-vector routing protocol, which is called a load-balanced routing protocol in this paper, for balancing load. The randomized scheme used LBR-BR1 to compute the value of the parameter *k* in order to exclude nodes that are too far away from the source from being candidates for an intermediate node. In this paper, we studied another method (LBR-BR2) to implement our routing approach. Both LBR-BR1 and LBR-BR2 are based on LBR-FR, which chooses an intermediate node from the entire network nodes. From our simulation results, we can see that LBR-BR2 has the best performance among the three LBR protocols as the number of connections increases. A direction of future work is to determine more dynamically the value of the parameter *k* for balancing network traffic load, improving network throughput and using a path of a distance as short as possible. For example, using the distance (or cost) from the particular destination for the parameter *k* may give better performance because then the intermediate node closer to the destination may be chosen.

8. Conclusions

We presented a family of load-balanced routing protocols to distribute the data traffic, via bounded randomization, over all available paths to a destination in the network for data load balancing. Our simulation results show that LBR-BR2 has improved performance with respect to throughput, message loss, message delay and link utilization at most times during the measurement interval, compared with LBR-FR, LBR-BR1, DVR and LSR, which are our previous load-balanced routing protocols and conventional destination routing protocols, respectively. Like the previous load-balanced schemes (LBR-FR, LBR-BR1), LBR-BR2 is simple and suffers from little control overhead. It also improves network performance with respect to throughput, message loss, message delay and link utilization. It randomly chooses one node within *k* hops as an intermediate node. This value of the parameter *k* dynamically changes according to the length of the shortest path from the source node to the destination node. Since our routing method may use a path that is more expensive than the shortest path, the gain from load balance may be smaller than the cost resulting from a longer delay in a network with a very balanced traffic. From the result of our simulation, we conclude that our new routing scheme has the following advantage over existing schemes:

Better utilization -- The improved network throughput, the lower packet loss and traffic load balancing achieved by using the proposed LBR schemes is indicative of better utilization than the shortest-path routing schemes.

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, pp. 78-84, San Antonio, Texas, Feb. 1999.

[4] 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.

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

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

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

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

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

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

[11] 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.

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

[13] 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.

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

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

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

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

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