

Performance Evaluation of 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 such as OSPF and RIP have the disadvantage of causing bottlenecks due to their inherent single-path routing. That is, the uniformly selected shortest path between a source and a destination may become highly congested even when many other paths have low utilization. We propose a family of routing schemes that distribute data traffic over the whole network via bounded randomization; in this way, they remove bottlenecks and consequently improve network performance. For each data message to be sent from a source s to a destination d , each of the proposed routing protocols randomly choose an intermediate node e from a selected set of network nodes, and routes the data message along a shortest path from s to e . Then, it routes the data message via a shortest path from e to d . Intuitively, we would expect that this increases the effective bandwidth between each source-destination pair. Our simulation results indicate that the family of proposed load-balanced routing protocols distribute traffic evenly over the whole network and, in consequence, increases network performance with respect to throughput, message loss, message delay and link utilization. Moreover, implementing our scheme requires only a simple extension to any shortest-path routing protocol.

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 *shortest-path routing (destination routing)*. In source routing, a source node determines the path that a data message must take [11]. In shortest-path 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 a shortest path to the destination.

Shortest-path routing protocols are classified into two types of routing protocols: *distance-vector routing* [17], for example, used in the RIP Internet protocol [13], and *link-state routing* [14], for example, used in the OSPF Internet protocol [15].

In the distance-vector routing protocol, each node maintains a routing table and a distance vector, which contain, respectively, a preferred neighbor for the shortest path to each destination in the network and the distance of the path to the destination. Each node has incomplete knowledge of the network topology and knows only its neighboring nodes. From these neighbors, the node chooses the closest neighbor to each destination. Each node periodically sends its distance vector to each of its neighbors to inform it of any distance changes to any destinations. The node determines which neighbor is the closest to each destination by comparing the distance vectors of its neighbors ([13], [17]).

Link-state routing protocols require each participating node to maintain complete network topology information. Each node actively tests the status of the links between itself and its neighbors. Then, it periodically broadcasts the local link status information to all other nodes. Since each node receives the local link status information from all other nodes, it is able to build a graph of the whole network topology and to compute the shortest path from itself to every other node ([14], [15]).

Shortest-path 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 remain 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 buffer overflow at some 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 any link 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 might 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 [8], 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, may increase the effective bandwidth up to the capacity of the minimum cut separating these two nodes.

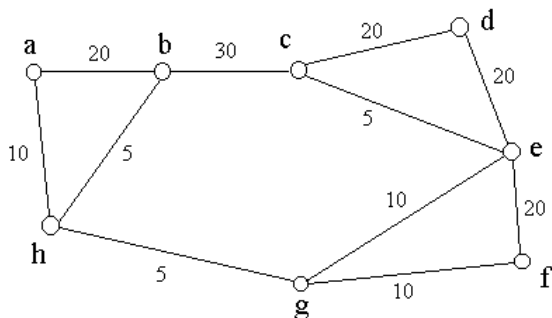


Figure 1. Network topology

For example, let's consider Figure 1. The number by each bi-directional 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 (length) 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. Now we can see that there exists an unused effective bandwidth between each pair of nodes in a shortest-path routing protocol, which we could use productively.

Several multiple-path routing techniques have been proposed to increase the effective bandwidth between each pair of nodes and to attempt thereby to improve performance ([2], [9], [18], [20], [21]). 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. We mention Shortest Path First with Emergency Exits [20] based on link-state routing, Multiple Disjoint Paths [18] 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 significantly increase the complexity of the routing algorithms. Several randomized multiple-path routing schemes ([7], [16], [19]) 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 load-balanced 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 flat network topology.

We proposed 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 [4]. We applied our routing algorithm to a hierarchical network topology [5].

In this paper, we simulate a family of our proposed routing protocols ([3], [4]) more comprehensively. We increase the number of connections in our simulation network topology up to $n \cdot (n-1)$, where n is the number of the nodes in the entire network.

The rest of this paper is organized as follows. Section 2 gives an overview of the max-flow/min-cut theorem. Section 3 sketches the load-balanced routing protocol. Section 4 introduces the protocol notation to give a formal version of our routing protocol, which is given in Section 5. In Sections 6 and 7, we present the simulation model and our results. Sections 8 and 9 outline future work and draw conclusions, respectively.

2. The Max-Flow/Min-Cut Theorem

We sketch the max-flow/min-cut theorem in this section [8].

2.1 Networks and Flows

A network $G(N, A)$ is defined to be a set N of nodes and a set A of pairs of distinct nodes from N called *arcs*. An arc (i, j) is an ordered pair, to be distinguished from the pair (j, i) . A path P in a network is a sequence of nodes (n_1, n_2, \dots, n_k) with $k \geq 2$ and a corresponding sequence of $k - 1$ arcs such that the i th arc in the sequence is (n_i, n_{i+1}) . Nodes n_1 and n_k are called the source node and the destination node, respectively. Each arc (i, j) is assigned a non-negative number $c(i, j)$, called the capacity of the arc (i, j) . A *flow* x_{ij} is the rate of traffic transmitted on the arc (i, j) in the network. For every arc (i, j) , $0 \leq x_{ij} \leq c(i, j)$.

2.2 Maximum Flow and Minimum Cut in a Network

The total flow F of a node is the sum of the flows into the node. In the max-flow problem, two nodes are distinguished: the source (s) and the destination (d). The objective is to push as much flow as possible from s to d while observing the capacity constraints.

Let S be a subset of nodes such that $s \in S$ and $d \notin S$. $N - S$ is the complement of S . Let $[S; N - S]$ be the set of arcs from any node in S to any node in $N - S$. $[S; N - S]$ is called the cut Q determined by S . Deletion of $[S; N - S]$ destroys all directed paths from s to d . Let us denote by $c(S)$ the capacity of the cut determined by S which is defined as follows:

$$c(S) = \sum_{a \in [S; N - S]} c(a)$$

There exist cuts between node s and node d in a network. A set with the minimum capacity among all sets of arcs in a network whose deletion destroys all directed paths from s to d is called *the minimum cut*.

Figure 2 shows an example of a cut and its capacity in a network D with the set of nodes $N = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}, \mathbf{f}\}$, given the source node \mathbf{a} and the destination node \mathbf{e} . We assumed that each arc has a capacity of 1. Let S be $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$; then $N-S$ is $\{\mathbf{d}, \mathbf{e}, \mathbf{f}\}$. Now we can see that $Q = \{(\mathbf{b}, \mathbf{d}), (\mathbf{a}, \mathbf{e})\}$ and $c(S) = 2$. In fact, the capacity of the minimum cut is 2.

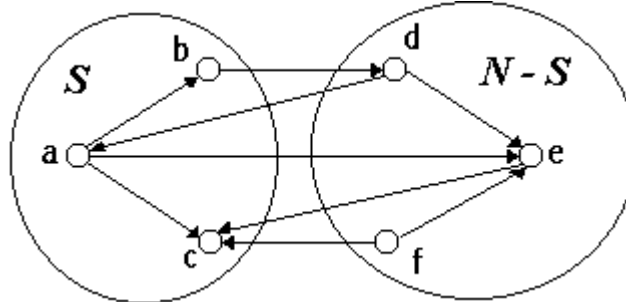


Figure 2. Illustration of a cut $Q = [S, N - S]$, where $S = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, source is node \mathbf{a} , destination is node \mathbf{e} and $Q = \{(\mathbf{b}, \mathbf{d}), (\mathbf{a}, \mathbf{e})\}$ and $c(S) = 2$.

For every pair of nodes s and d , with total flow F into d , and every S , $F \leq c(S)$. The Max-Flow/Min-Cut theorem can now be stated as follows:

Max-Flow/Min-Cut Theorem [8]: *Every network has a maximum total flow F , which is equal to the capacity of the minimum cut.*

3. Overview of the Load-Balanced Routing

In this section, we informally 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 (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. In order to distribute data

traffic over the entire network, we select an intermediate node and route a data message from source to destination through a selected intermediate node. Our scheme is based on a shortest-path routing algorithm. Here is the basic idea of LBR:

1. *LBR selects a set S of nodes from the network nodes.*
2. *For each data packet to be sent from a source node s to a destination node d , the proposed routing scheme randomly chooses an intermediate node e among the nodes in S .*
3. *LBR routes the packet via the shortest-distance (or least-cost) path from s to e .*
4. *Then, LBR routes the packet via the shortest-distance (or least-cost) path from e to d .*

To accomplish this, each message must carry at least three pieces of information: the destination d , the intermediate node e , and a bit b . The 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 to 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 many more paths between a source and a destination in the network than the non-randomized routing schemes 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 [8].

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 a set of 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**.

The question to be considered next is how to select a set of candidates for the intermediate node.

3.1 Load-Balanced Routing via Full Randomization

We can use simply all the network nodes for a set of candidates for an intermediate node. In this case, our routing scheme is called Load-Balanced Routing via Full Randomization (LBR-FR). LBR-FR may increase the effective bandwidth between each source-destination pair up to the capacity of minimum cut separating the pair, which is the upper bound on the effective bandwidth [8].

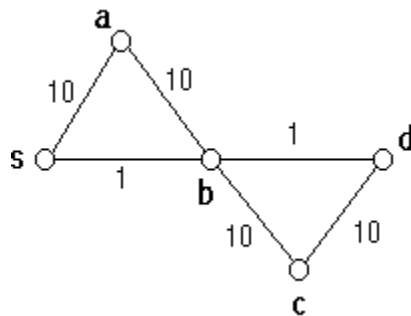


Figure 3. A network topology

Figure 3 shows a case where LBR-FR cannot use all the paths available between each source-destination pair, even when LBR-FR uses all the nodes in the network for a set of candidates for an intermediate node. There is a path **s-a-b-c-d** between the source **s** and the destination **d**, but LBR-FR will not use the path to route data packets from **s** to **d**. Even with full randomization of choosing an intermediate node from all the nodes in the network, LBR-FR will get only 1 as the effective bandwidth between the source **s** and the destination **d**, even though the minimum capacity between the pair is 11.

LBR-FR 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, 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 scheme results in the use of a path of length 6.

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

3.2 Load-Balanced Routing via Bounded Randomization

To remedy the above mentioned problem of LBR-FR, we introduce a parameter k , with the goal of excluding nodes that are “too far away” from the source from being candidates for an intermediate node. The value of the parameter k will be a distance from the source node. 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 a conventional shortest-path 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 generally will alleviate congestion and increase 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 3.1). 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 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 may

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]:

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

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. This static value of k may be 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 a static value for the parameter k , we may choose the value of the parameter k dynamically so that the distance between source and destination is factored in (LBR-BR2) [4]:

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

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 changes dynamically according to the length of the shortest path from the source node s to the destination node d . Since the factor of $\text{dist}(s, d)$ is less than 1, the length of any path chosen in this way is strictly less than three times the length of the shortest path (which is of course $\text{dist}(s, d)$), assuming that the distance function behaves like a true metric.

4. Protocol Notation

In this paper, we use a simple notation to define our routing protocol. A protocol is defined by a set of processes, $p[0], p[1], \dots, p[n-1]$. 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 ";".

A process is executed by first computing the current Boolean value of the guard of each action, then selecting arbitrarily one guard whose value is true and executing its corresponding statement. 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. Enabled actions in the same process or in different processes in a network are executed one at a time (Action Atomicity). Any action that is enabled at a network state can be selected for execution at the state (Nondeterministic Execution). 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 (Maximal Execution). Executions are assumed to be fair, i.e., each action that remains continuously enabled is eventually executed (Action Fairness).

The communication between processes is based on a message-passing model. For every pair of neighboring processes $p[i]$ and $p[j]$, we assume 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 [9] and [12].

5. 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 $p[i]$. The 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 [$i: 0 \dots n-1$]

constants

n : **integer** {number of nodes in the network}

inputs

N : **set of** $\{j \mid p[j] \text{ is a neighbor of } p[i]\}$

variables

k : $0 \dots n-1$, {maximum length of 1st routing path}

$inter$: **set of** $\{0 \dots n-1\}$ {possible intermediate nodes of routing paths}

rtb : **array** $[0 \dots n-1]$ **of** $0 \dots n$, {routing table}

hop : **array** $[0 \dots n-1]$ **of** $0 \dots n$, {hop count to each destination}

h : **array** $[0 \dots n-1]$ **of** $0 \dots n$, {neighbor's hop count to each destination}

e, d : $0 \dots n-1$, {message's intermediate node and destination}

b : $0 \dots 1$, {status bit: $b=0$ on 1st routing path, $b=1$ on 2nd routing path}

x : $0 \dots n-1$,

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 [i, b, e, d, n]
  [] 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 [i, j, h, n]
end

```

In the first action, process $p[i]$ computes k as the average hop count to each reachable destination in the network. Then, the set of candidate nodes for intermediate routing nodes is computed. In the second action, the process creates a data message and chooses a destination for the message. Also, an intermediate node is chosen at random from the set of intermediate candidates. Then, the created data message is routed to a neighboring process using statement RTMSG, defined below.

In the third action, the process receives a data message and then routes the message to a neighbor using statement RTMSG. In the fourth action, the process sends a copy of its distance vector to each of its neighbors. In the last action, the process receives a copy of the distance vector from one of its neighbors, and updates its routing table and local distance vector using statement UPDTBL, which is defined below.

Statement RTMSG [i, b, e, d, n] is defined as follows.

```

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

```

```

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

```

In this statement, the process checks fields b, e, and d, of the message. If $d = i$, the message has reached its destination and is delivered. Otherwise, if $b = 0$, then the message is in its first routing path, and it is routed towards process $p[e]$. If $b = 1$, the message is in its second routing path, and it is routed towards process $p[d]$.

Statement UPDTBL [i, j, h, n] is defined as follows.

```

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

```

In the first case of this statement, if process $p[i]$ currently routes to a destination $p[x]$ through the neighbor $p[j]$, and $p[j]$'s distance to the destination changes, process $p[i]$ updates its distance to $p[x]$ accordingly. In the second case, if the neighbor $p[j]$ uses a shorter path to reach a destination $p[x]$, then process $p[i]$ chooses $p[j]$ as the next hop to destination $p[x]$. In the third case, process $p[j]$ does not have a shorter path to $p[x]$ than $p[i]$'s path, and thus, $p[i]$ does not

update its tables. In the last case, the next hop neighbor to $p[x]$ is no longer connected to $p[i]$ due to a failure, and thus $p[i]$ chooses $p[j]$ as its next hop to $p[x]$.

6. 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 used in our simulations are DVR, LSR, LBR-FR, LBR-BR1, and LBR-BR2. DVR is a distance-vector and loop-free routing protocol [17], which uses a shortest-distance path for each pair of source and destination nodes. LSR is a link-state routing protocol [14], where each node calculates and broadcasts the costs of its outgoing links periodically and Dijkstra's shortest path algorithm [10] is applied to the view of the network topology to determine next hops. To get a better understanding of the proposed LBR protocols, we compare the performance of our three LBR protocols against that of DVR and LSR.

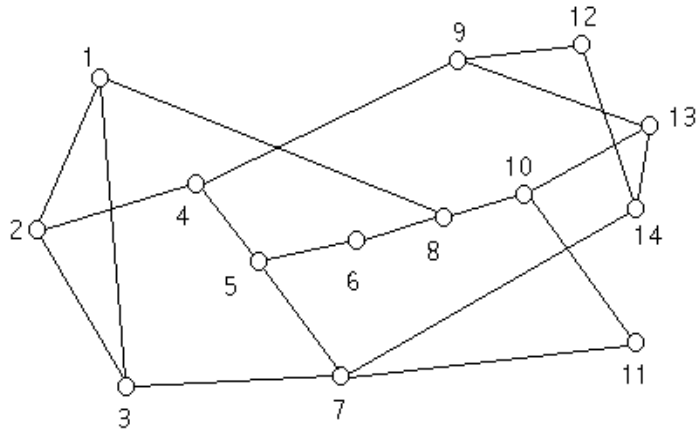


Figure 3. NSFNET Topology: 14 nodes, 21 bi-directional links, average degree 3

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 μ sec. 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 acknowledgments 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.

7. Simulation Results

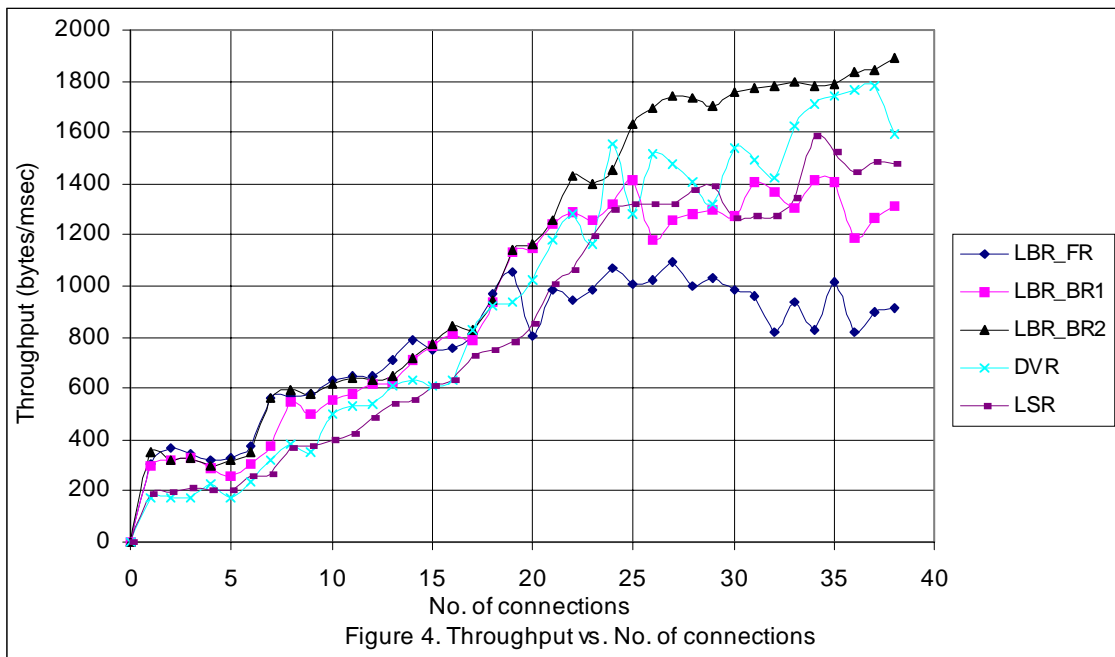


Figure 4 shows throughput versus the number of connections. The throughput in all the routing protocols in general increases as the number of connections increases. With respect to throughput, the three LBR protocols are better than DVR and LSR, when the number of connections is low. The throughput of 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.

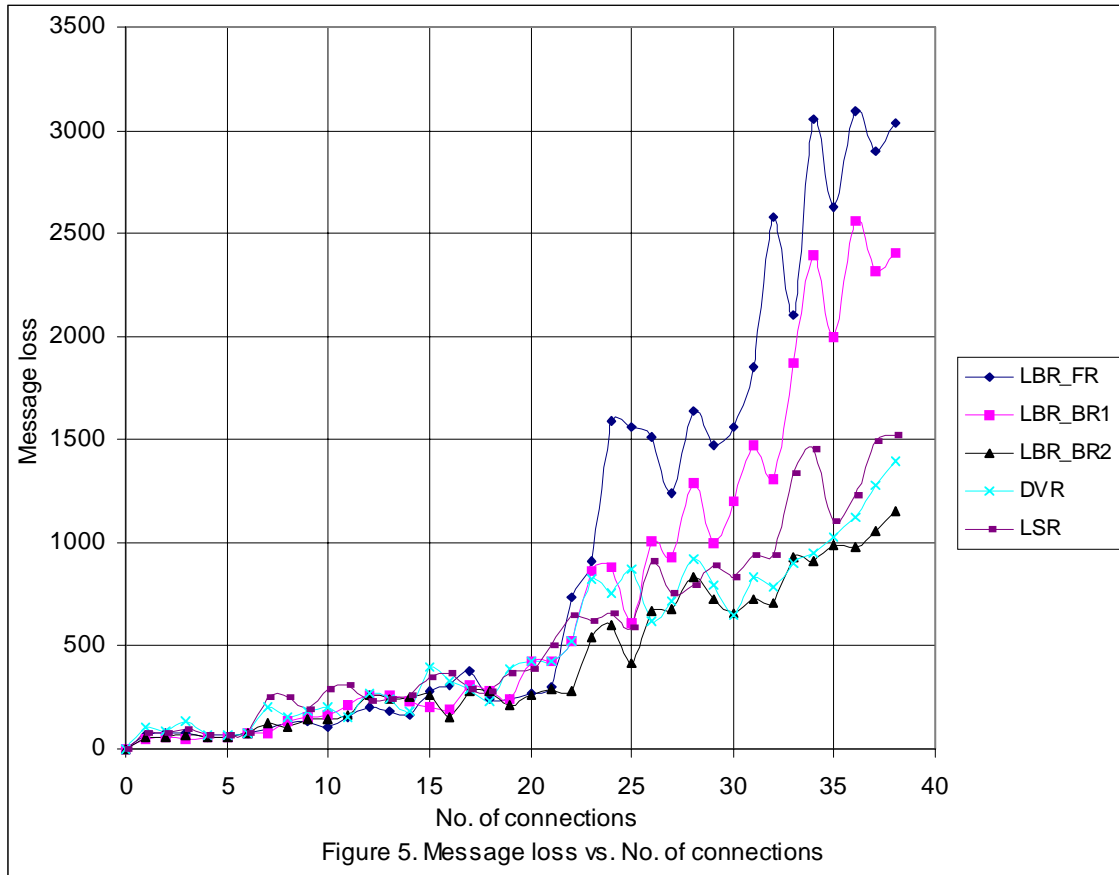


Figure 5 shows the message loss versus the number of connections. The message loss in the LBR-BR2 protocol is generally lower than in the other routing protocols both when the numbers of connections are low and high. LBR-FR and LBR-BR1 have higher message loss when the number of connections is in the range from 23 to 38, while LBR-BR2 has a message loss lower than DVR and LSR when the number of connections is in the range from 23 to 38.

Figure 6 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 in the range from 1 to 21. 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.

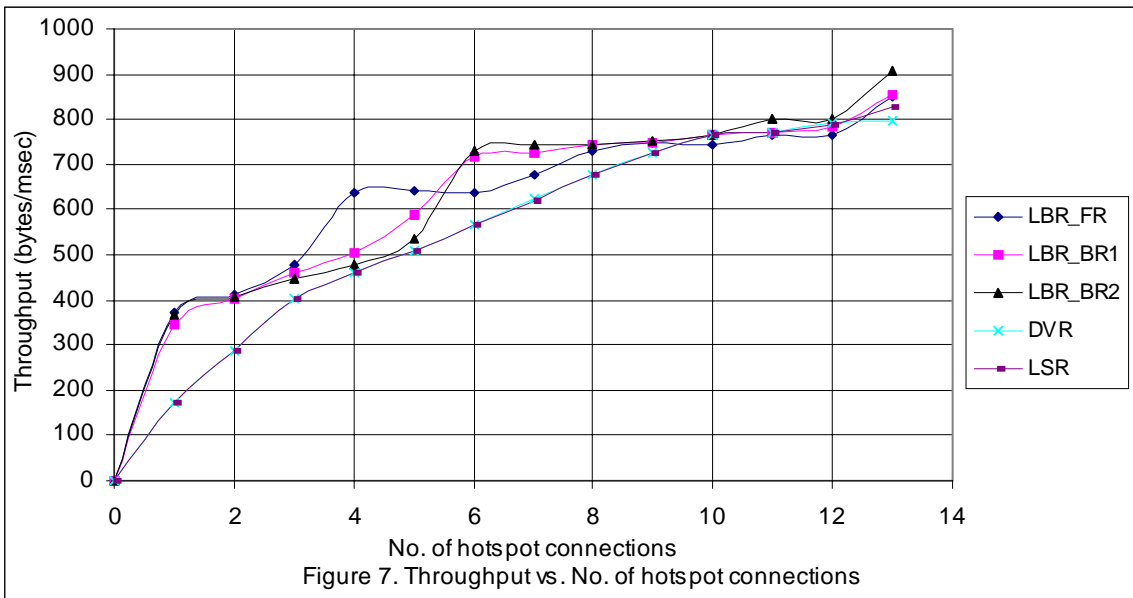
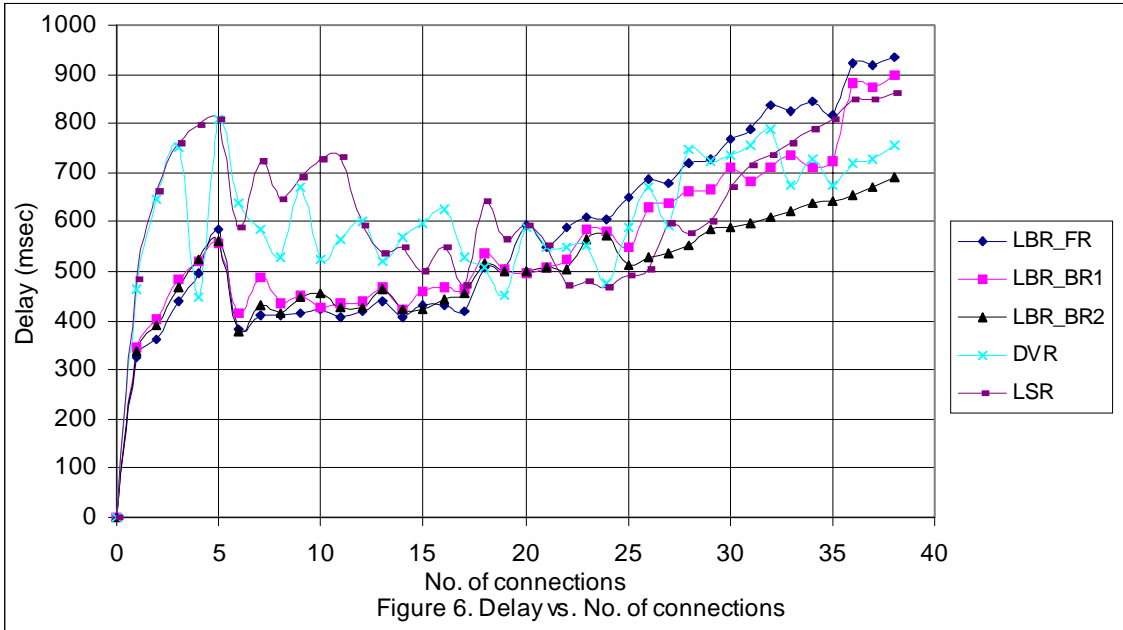


Figure 7 shows throughput versus number of connections in the hot spot. In this scenario, a special node, called hotspot, has many more connections than the other nodes in the network. All the LBR protocols have better throughput than DVR and LSR do until the number of hotspot connections is 9. LBR-FR 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.

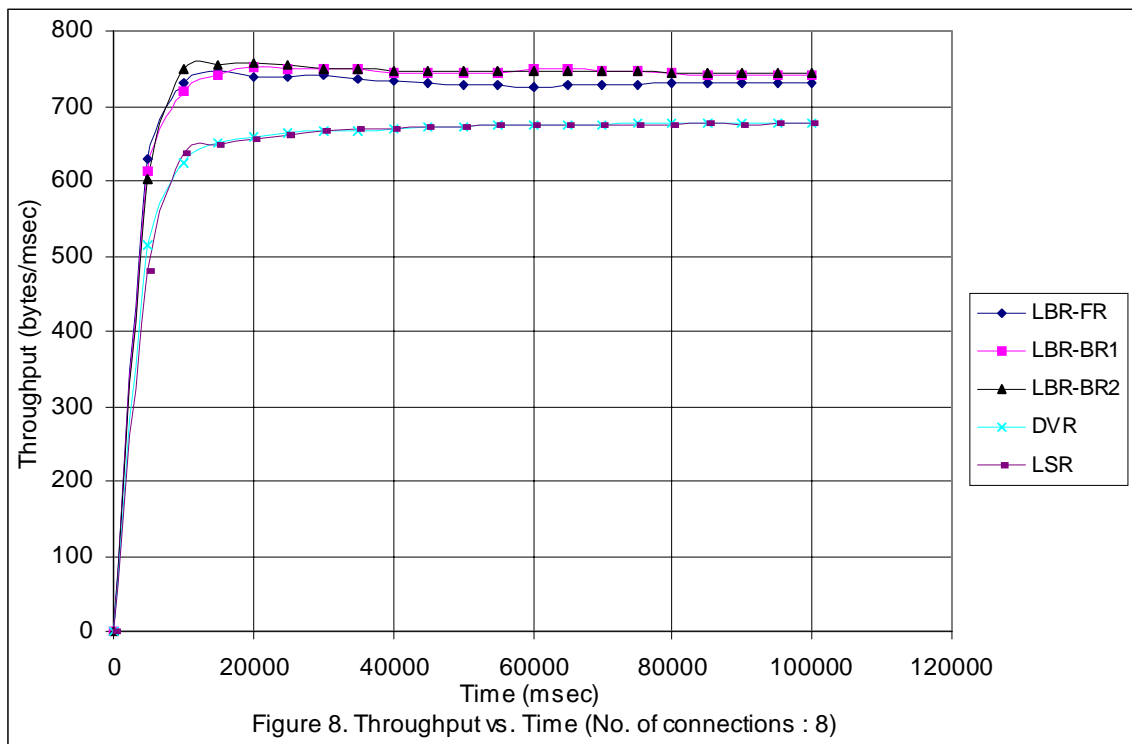


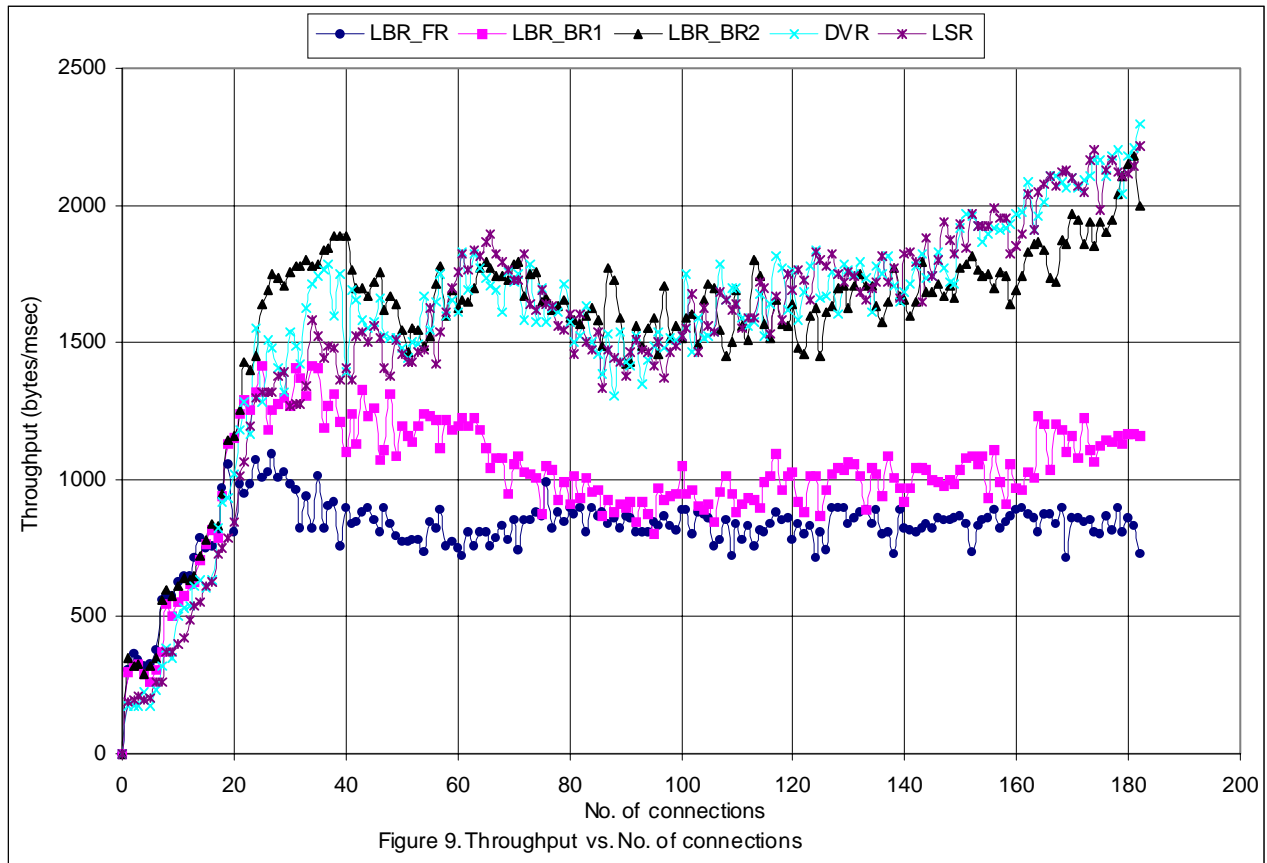
Figure 8 shows throughput versus time when the number of connections in the hotspot is 8. DVR and LSR have about the same network throughput during the measurement interval. DVR and LSR have lower throughput than the LBR protocols at all times during the measurement time. LBR-BR2 has better throughput than the other LBR protocols at all times during the measurement time.

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 4).

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

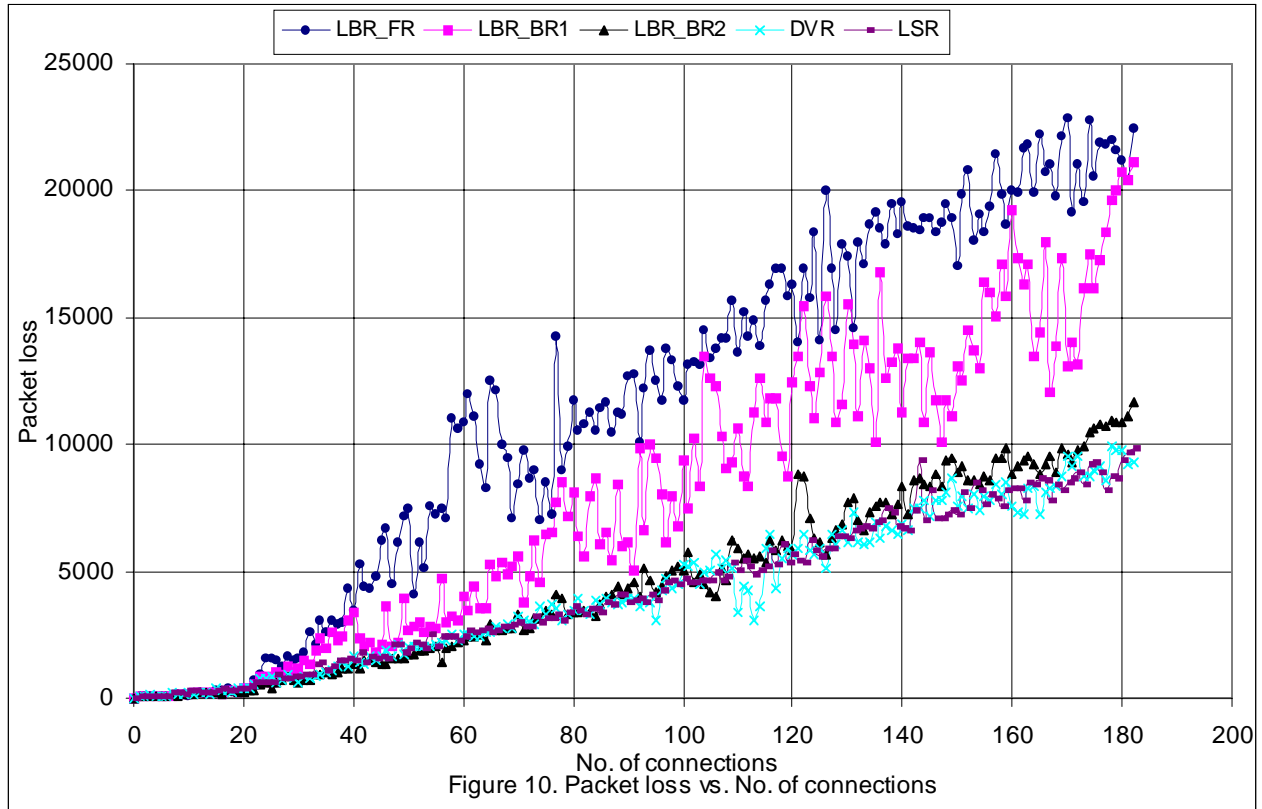
Table 1. Average link utilization (No. of connections = 8)

In Figures 9 and 10, we increased the number of connections in our simulation network topology up to $n \cdot (n-1)$, where n is the number of the nodes in the entire network. Our network topology consists of 14 nodes; thus, $n \cdot (n-1) = 182$. In both throughput and message loss versus the number of connections, the proposed routing protocols outperform the LSR and DVR protocols when the number of connections is low. The curves tend to converge as the number of connections increases (especially when the number of connections is in the middle). The proposed routing protocols have a performance worse than the LSR and DVR when the number of connections is high.



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. 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); however, it exhibits the worst performance among the considered routing protocols when the number of connections is high (in the range from 20 to 182). LBR-FR will increase the effective bandwidth up to the capacity of the minimum cut separating these two nodes, but it has a much better chance than LBR-BR1 and LBR-BR2 that a data message is routed to the destination via a path that is much longer than a shortest path from the source to the destination. LBR-BR1 also exhibits a good performance with respect to throughput and message loss when the number of connections is low (in the range from 1 to 25), while it exhibits a performance worse than DVR and LSR when the number of connections is high (in the range from 34 to 182). LBR-BR1 will decrease the effective bandwidth less than the capacity of the minimum cut separating these two nodes, while it reduces the chance that a data message is routed to the destination via a path that is much longer than a shortest path from the source to the destination. LBR-BR2 exhibits good

performance with respect to throughput and message loss when the number of connections is in the range from 1 to 100. The performance of the shortest-path routing protocols (DVR and LSR) is getting better as the number of connections increases. In particular, DVR and LSR exhibit better performance with respect to throughput and message loss when the number of connections approaches 182. 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 $n \cdot (n-1)$, where n is the number of the network nodes.

8. Future Work

We proposed a family of load-balanced routing protocols for balancing load. In contrast to LBR-FR, LBR-BR1 and LBR-BR2 use a 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 the proposed load-balanced schemes and compared them with conventional shortest-path routing protocols. From our simulation results, we see that LBR-BR2 has the best performance among all the routing protocols as the number of connections is in the range of from 1 to 80. The curves tend to converge as the number of connections approaches 90. From this point on, the performances of DVR and LSR are getting better than those of the load-balanced routing protocols. A direction of future work is to determine the value of the parameter k to delay the cross point of the curves. Specifically, we will try to determine the value of the parameter k to improve the performance of our schemes when the number of connections is high. For example, using the distance (or cost) from the particular destination for the parameter k may give better performance because then an intermediate node closer to the destination may be chosen.

9. 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 one of the proposed schemes (LBR-BR2) has good performance with respect to throughput, message loss, message delay and link utilization, compared with DVR and LSR, which are conventional destination routing protocols. The LBR schemes are simple and suffer from little control overhead. LBR-BR2 has the best performance when the number of connections is in the range of 1 to 80. 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 scheme 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 very balanced traffic. From the results of our simulation, we conclude that our new routing scheme (especially LBR-BR2) has the following advantage over existing schemes except when the number of connections approaches $n \cdot (n-1)$:

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 that of the conventional 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, San Antonio, Texas, Feb. 1999.
- [4] S. Bak, J. A. Cobb, E. L. Leiss, *Load-Balanced Routing via Randomization*, Proceedings of CLEI, Asuncion, Paraguay, Aug. 30, 1999.
- [5] S. Bak, J. A. Cobb, E. L. Leiss, *Hierarchical Load-Balanced Routing via Bounded Randomization*, Proceedings of The Eighth IEEE International Conference on Computer Communications and Networks, pp. 589-594, Cambridge, MA, Nov. 3-6, 1999.
- [6] S. Bak, J. A. Cobb, E. L. Leiss, *Load-Balanced Routing via Bounded Randomization*, Proceedings of The eleventh IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 857-862, Boston, MA, Oct. 11-13, 1999.
- [7] R. Cole, B.M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A.W. Richa, K. Schroeder, R.K. Sitaraman, and B. Voeking, *Randomized Protocols for Low-Congestion Circuit Routing in Multistage Interconnection Networks*, Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 378-388, May 1998.
- [8] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.
- [9] J. A. Cobb and M. G. Gouda, *Balanced Routing*, IEEE Proceedings of the International Conference on Network Protocols, 1997.
- [10] E. W. Dijkstra, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik, Vol. 1, pp. 269- 271, 1959.
- [11] R. C. Dixon, D. A. Pitt, *Addressing, Bridging, and Source Routing (LAN Interconnection)*, IEEE Network, Vol. 2, No. 1, Jan.1988.
- [12] M. Gouda, *The Elements of Network Protocol Design*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., 1998.
- [13] G. Malkin, RIP Version 2, *Internet Request for Comments 1723*, Nov. 1994, Available from <http://www.ietf.cnri.reston.va.us>.

- [14] J.M. McQuillan, Ira Richer and E.C. Rosen, *The New Routing Algorithm for the ARPANET*, IEEE Trans. on Communications, Vol. COM-28, NO. 5, pp. 711-719, May 1980.
- [15] J. Moy, *OSPF Version 2*, Internet Request For Comments 2178, July 1997. Available from <http://www.ietf.cnri.reston.va.us>.
- [16] 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.
- [17] Segall and M. Sidi, *A Failsafe Distributed Protocol for Minimum Delay Routing*, IEEE Trans. on Commun., COM-29(5), 686-695, May 1981.
- [18] D. Sidhu, R. Nair and S. Abdallah, *Finding Disjoint Paths in Networks*, Proceedings of the 1991 ACM SIGCOMM Conference, 1991.
- [19] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.
- [20] Z. Wang and J. Crowcroft, *Shortest Path First with Emergency Exits*, Proceedings of the 1990 ACM SIGCOMM Conference, 1990.
- [21] 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.