

# Hierarchical Load-Balanced Routing via Bounded Randomization

*Sangman Bak*

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

*Jorge A. Cobb*

Dept. of Computer Science  
University of Texas at Dallas  
Dallas, TX 75083-0688  
jcobb@utdallas.edu

*Ernst L. Leiss*

Dept. of Computer Science  
University of Houston  
Houston, TX 77204-3475  
coscel@cs.uh.edu

**Abstract-** The purpose of routing protocols in a computer network is to maximize network throughput. Popular 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. In this paper, we propose a routing scheme for hierarchically structured computer networks such as Internet. The new hierarchical routing algorithm, which we call the Hierarchical Load-Balanced Routing (HLBR), is based on distance-vector routing algorithms. HLBR randomly balances traffic load over the whole network; 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 chooses randomly each of three intermediate nodes from a selected set of network nodes, and routes the data message along a path from  $s$  through the three intermediate nodes to  $d$ . This increases the effective bandwidth between each pair of nodes.

## 1. INTRODUCTION

In any wide-area store-and-forward computer network, such as the Internet, routing protocols are essential. They are responsible for finding an efficient path between any pair of source and destination nodes in the network, and routing data messages along this path. The path must be chosen so that message delay, message loss and other undesirable events are minimized.

In the past, most of the works in network routing have been focused on Shortest-path routing protocols. Unfortunately, these Shortest-path 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 mechanisms for altering the routing other than updating the routing tables. Routing table updates occur too slowly to respond to significant traffic fluctuations. Furthermore, increasing the frequency of routing table updates leads to unstable network behavior and an increase in the network load due to routing state messages [6], [18].

Note that the shortest path may be highly congested, even when many other paths to the destination are not congested. This congestion may trigger the loss of valuable messages due to buffer overflow at some node [26]. Moreover, using the same 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 algorithms and networks. A result in network flow theory, known as the max-flow min-cut theorem [7], shows that distributing the traffic load over the available paths between a source and a destination in the network, instead of using only one path of the minimum cost, increases the maximum possible throughput up to the capacity of the minimum cut separating these two nodes.

For example, let's consider Figure 1. The number beside each link represents its capacity. Suppose that node **a** (source) wants to send the data messages to node **f** (destination). Suppose that we use the hop count in order to calculate the cost of a path in the network. Then the effective bandwidth (i.e. the capacity of the minimum cut)

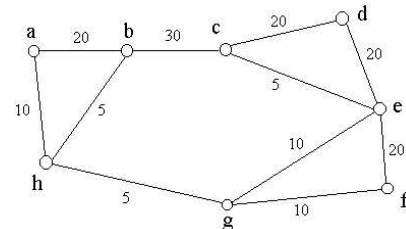


Figure 1. Network topology

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.

So far, 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], [8], [18], [23], [25], [26]). These routing protocols improve performance by routing data messages via multiple paths to the destination. They maintain a stable routing table and meanwhile provide alternate paths to distribute data traffic when it is accumulating in some nodes of the network. When congestion and network failures do occur, instead of initiating route updating, the source temporarily forwards the traffic along the alternate paths and passes around the congested areas or failed areas. If the changes are persistent, the routing tables will be updated eventually when the next route updating time is due. Some multiple-path routing techniques are Shortest Path First with Emergency Exits [25] based on link-state routing ([16], [17]), Multiple Disjoint Paths [23] based on distance-vector routing ([11], [15], [22]), and Dynamic Multi-path Routing [2] based on source routing [10]. But these techniques require considerable processing overhead, need non-negligible storage space, or significantly increase the complexity of the routing algorithms. Several randomized multiple-path routing schemes ([5], [20], [24]) have been proposed for regular network topologies of parallel computers, 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 multi-path routing scheme, which is called Load-Balanced Routing (LBR), to address the bottlenecks associated with Shortest-path routing protocols. The routing protocol was formulated for an IP (Internet Protocol) network with an irregular topology. We showed that the randomized scheme increased throughput in the network with a flat network addressing structure. However, the nodes in the Internet are organized into a hierarchical structure.

There have been several proposals for hierarchical routing ([1], [4], [19], [21]), which vary in the way in which the nodes are organized and the routing schemes are used. With very few exceptions, existing routing protocols for hierarchically structured Internet have been based on single-path routing protocols.

In this paper we extend LBR to a hierarchical routing scheme based on bounded randomization that improves the network throughput by distributing the traffic load over all available paths to the destination in the network. In consequence, network throughput is increased and message loss is reduced.

The rest of this paper is organized as follows. Sections 2 and 3 present the distance-vector routing protocol and the load-balanced routing protocol, respectively. In Sections 4 and 5, we present the assumed two-level hierarchical network model and a hierarchical routing protocol, respectively, which our protocol is based on. Section 6 presents the overview of the hierarchical load-balanced routing protocol. Section 7 introduces the protocol notation to give a formal version of our routing protocol, which is given in Section 8. In Section 9, we present future work and draw conclusion.

## 2. DISTANCE-VECTOR ROUTING PROTOCOL

HLBR is based on a distance-vector routing protocol [22]. In a distance-vector routing algorithm, 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. At a stable state of the protocol, the path made by consecutive preferred neighbors for a given destination is the shortest path to the destination. Each node periodically sends its distance vector to each of its neighbors to inform it of any distance changes to any destinations. By using Dijkstra's shortest path algorithm [9], the node determines which neighbor is the closest to each destination by comparing the distance vectors of its neighbors.

## 3. LOAD-BALANCED ROUTING

In this section, we sketch how LBR protocol routes data messages to the destination. LBR was designed for a network with a flat network structure [3]. Each node should forward data messages to its neighbors so that the cost of the 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. LBR is based on the distance-vector routing algorithm [22].

### 3.1 Simple Load-Balanced Routing

In this subsection, we sketch Simple Load-Balanced Routing (SLBR) with ideas borrowed from Valiant's randomized routing algorithm [24]. 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 Simple Load-Balanced Routing**:

1. For each data packet to be sent from a source node  $s$  to a destination node  $d$ , SLBR 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. Then, it routes the packet via the shortest-distance (or least-cost) path from  $e$  to  $d$ .

This technique distributes the traffic load over many more paths between a source and a destination in the network than a single-path routing scheme 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 [7].

### 3.2 Load-Balanced Routing via Bounded Randomization

The protocol described in Subsection 3.1 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 chooses, at random, node  $f$  as the intermediate node. Then the proposed algorithm sends the data message to node  $f$  via the shortest path ( $a-h-g-f$ ) and then sends it to node  $b$  via the shortest path ( $f-e-c-b$ ). Although there is the path of length 1 between node  $a$  and node  $b$ , the SLBR may use a path of length 6. Clearly, routing paths that are excessively long will waste network resources.

To remedy the 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  $k$ .

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  $d_i$  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  $d$ .

## 4. TWO-LEVEL HIERARCHICAL NETWORK MODEL

In this section, we describe a two-level hierarchical network model, which we assume for the proposed routing protocol. The nodes of a network are clustered into  $m$  regions. Each region has at most  $n$  nodes. All the nodes that are within the same region are called *local nodes*. Nodes that have links to nodes in other regions are called *gateways*. Two regions are *neighbors* when there is one or more links between nodes in the two regions. In the network, each region is connected in the following sense. For any two nodes  $p$  and  $q$  in region  $i$ , there is a sequence of nodes  $p_0, \dots, p_r$  in region  $i$  such that  $p$  is  $p_0$ ,  $q$  is  $p_r$ , and for every  $k$ ,  $0 \leq k < r$ ,  $p_k$  and  $p_{k+1}$  are neighbors in the network. Each node in this network can be uniquely identified by two identifiers  $i$  and  $j$ . Identifier  $i$  indicates the region to which the node belongs. Identifier  $j$  indicates the node in region  $i$ .

In the proposed protocol, every node keeps its routing table. The routing table of each gateway consists of the following two small tables named *prs* and *rgn*:

- 1) *prs* is a local routing table with  $n$  entries, where  $n$  is the maximum number of nodes in the local region. Each entry  $j$  represents the cost of the path to the local node  $j$  and a preferred neighbor node along the path.
- 2) *rgn* is a global routing table with  $m$  entries, in which  $m$  is the maximum number of regions in the network. Each entry contains the cost of the path to a destination region and a preferred neighbor node along the path to the destination region.

Each local node keeps only its local routing table (*prs*).

## 5. HIERARCHICAL ROUTING PROTOCOL

In this section, we present in brief a hierarchical routing protocol, on which the proposed routing protocol is based.

In a network, when a data message is to be sent, the two identifiers (*dr*, *dp*) of its destination node are attached to the message before the message is sent. When a data (*dr*, *dp*) message arrives at a node, the node uses its routing table and the index (*dr*, *dp*), which identifies the message destination, to determine the preferred neighbor to which the message is forwarded.

In hierarchical routing, at least one node in each region is called a gateway of the region. The way a node  $p[i, j]$  routes a data(*dr*, *dp*) message depends on whether  $p[i, j]$  is a gateway of its own region  $i$ . When a data(*dr*, *dp*) message is received by node  $p[i, j]$ , the index (*dr*, *dp*) of the message destination is compared with the index ( $i, j$ ) of receiving node. If  $p[i, j]$  is not a gateway of region  $i$  and  $dr \neq i$ , then  $p[i, j]$  forwards the message towards a gateway of region  $i$ , using its *prs*. If  $p[i, j]$  is not a gateway of region  $i$  and  $dr = i$ , then the node  $p[i, j]$  forwards the message towards its ultimate destination using its *prs*. If  $p[i, j]$  is a gateway of region  $i$  and  $dr \neq i$ , then  $p[i, j]$  forwards the message towards a gateway of region  $dr$ , using its *rgn*. If  $p[i, j]$  is a gateway of region  $i$  and  $dr = i$ , then the node  $p[i, j]$  forwards the message towards its ultimate destination using its *prs*.

## 6. OVERVIEW OF HIERARCHICAL LOAD-BALANCED ROUTING

In this section, we sketch how our method, called Hierarchical Load-Balanced Routing (HLBR) works. The HLBR protocol is a combination of the distance-vector routing algorithm, LBR and hierarchical routing protocol given in the previous sections.

### 6.1 Simple Hierarchical Load-Balanced Routing

In this subsection, we first present a Simple Hierarchical Load-Balanced Routing (SHLBR).

Here is the algorithm of SHLBR:

1. For each data message to be sent from a source  $s$  to a destination  $d$ , SHLBR chooses randomly three intermediate nodes. The 1<sup>st</sup> intermediate node  $e1$  is chosen from the local nodes in the source region, the 2<sup>nd</sup> intermediate node  $e2$  is chosen from the gateways in the entire network and the 3<sup>rd</sup> intermediate node  $e3$  is chosen from the local nodes in the destination region.
2. It routes the data message along the shortest path from  $s$  to  $e1$  and then routes the data message along the shortest path from  $e1$  to the gateway in the source region.
3. It routes the data message via the shortest path from the gateway in the source region to  $e2$  and then via the shortest path from  $e2$  to the gateway in the destination region.
4. Finally it routes the data message via the shortest path from a gateway in the destination region to  $e3$  and then via the shortest path from  $e3$  to  $d$ .

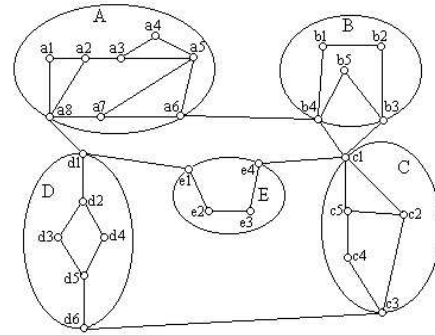


Figure 2. Network topology

As an example, consider Figure 2. Suppose that node  $a3$  wants to send the data messages to node  $c3$ . For each data message to be sent from  $a3$  to  $c3$ , SHLBR chooses randomly three intermediate nodes. The 1<sup>st</sup> intermediate node (say  $a5$ ) is chosen from the local nodes in source region A, the 2<sup>nd</sup> intermediate node (say  $b4$ ) is chosen from the gateways in the entire network and the 3<sup>rd</sup> intermediate node (say  $c5$ ) is chosen from the local nodes in the destination region. SHLBR routes the data message along the shortest path ( $a3$ - $a5$ ) from  $a3$  to  $a5$  and routes the data message along the shortest path ( $a5$ - $a6$ ) from  $a5$  to the gateway  $a6$  in the source region A. Then, it routes the data message via the shortest path ( $a6$ - $b4$ ) from  $a6$  to  $b4$  and then via the shortest path ( $b4$ - $c1$ ) from  $b4$  to the gateway  $c1$  in destination region C. Finally it routes the data message via the shortest path ( $c1$ - $c5$ ) from  $c1$  to  $c5$  and then via the shortest path ( $c5$ - $c4$ - $c3$ ) from  $c5$  to  $c3$ .

### 6.2 Bounding the Randomization of the Routing Path

The protocol of the previous subsection, however, has a weakness. It is possible that a data message is routed to the destination via a very long path, much longer than the shortest path between the source and the destination.

For example, in Figure 2, suppose that node  $a4$  in the region A wants to send a data message to the node  $b1$  in the region B and it randomly chooses nodes  $a1$ ,  $e4$  and  $b2$  as the intermediate nodes. SHLBR routes the data message along the shortest path ( $a4$ - $a3$ - $a2$ - $a1$ ) from  $a4$  to  $a1$  and routes the data message along the shortest path ( $a1$ - $a8$ ) from  $a1$  to the gateway  $a8$  in the source region A. Then, it routes the data message via the shortest path ( $a8$ - $d1$ - $e1$ - $e4$ ) from  $a8$  to  $e4$  and then via the shortest path ( $e4$ - $c1$ - $b3$ ) from  $e4$  to the gateway  $b3$  in destination region B. Finally it routes the data message via the shortest path ( $b3$ - $b2$ ) from  $b3$  to  $b2$  and then via the shortest path ( $b2$ - $b1$ ) from  $b2$  to  $b1$ . Although there is the path of length 4 between node  $a5$  and node  $b1$ , SHLBR may use the path of length 11. Clearly, routing paths that are excessively long will waste network resources.

To remedy this problem, we introduce three parameters  $k1$ ,  $k2$  and  $k3$ , in order to exclude the nodes which are too far away from the source from being candidates for the 1<sup>st</sup> intermediate node, from the source gateway from being candidates for the 2<sup>nd</sup> intermediate node and from destination gateway from being candidates for the 3<sup>rd</sup> intermediate node, respectively. The set of candidates for the 1<sup>st</sup> intermediate node is restricted to all those nodes whose distance to the source node is at most  $k1$ . The set of candidates for the 2<sup>nd</sup> intermediate node is restricted to all those gateways whose distance to the gateway in the source region is at most  $k2$ . The set of candidates for the 3<sup>rd</sup> intermediate node is restricted to all those nodes whose distance to the gateway in the destination region is at most  $k3$ .

The values chosen for these  $k_1$ ,  $k_2$  and  $k_3$  affect delay, the traveled path length, load balancing and network throughput. Choosing small values will increase the likelihood of bottleneck. On the other hand, choosing large values will waste network resources by routing messages via excessively long paths. To reach a compromise between these two extremes, we may choose  $k_1$ ,  $k_2$  and  $k_3$  to be the average of the distance to each node in the source region reachable from the source node, the average of the distance to each gateway in the entire network reachable from the gateway in the source region, and the average of the distance to each node in the destination region reachable from the gateway in the destination region, respectively.

## 7. PROTOCOL NOTATION

In this paper, we use a simple notation to define our routing protocol. A protocol is defined by a set of processes,  $p[v]$ , where  $v$  is in  $\{(i, j) \mid i: 0..m-1; j: 0..n-1\}$ . A process corresponds to a node in a computer network. A pair of neighboring processes is joined with 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  $\parallel$  as follows.

**begin** *action.1*  $\parallel$  *action.2*  $\parallel$  ...  $\parallel$  *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 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  $p[u]$  and  $p[v]$ , where  $u$  and  $v$  are in  $\{(i, j) \mid i: 0..m-1; j: 0..n-1\}$ , there are a FIFO channel from  $p[u]$  to  $p[v]$  and a FIFO channel from  $p[v]$  to  $p[u]$ . The statement **send** *data*(*var*) **to**  $p[v]$  in process  $p[u]$  appends a message of type *data* to the channel from  $p[u]$  to  $p[v]$ , and the field in the message is the current value of variable *var* in process  $p[u]$ .

In addition to Boolean expressions, guards in each process  $p[u]$  are allowed to be of the form **rcv** *data*(*var*) **from** **any**  $p[v]$ . This guard is enabled if and only if there is a message of type *data* at the head of an incoming channel of  $p[u]$ . 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 [12] and [13].

## 8. SPECIFICATION OF HIERARCHICAL LOAD-BALANCED ROUTING PROTOCOL

In this section, we present a formal version of the hierarchical load-balanced routing protocol. Each process has constants  $m$  and  $n$  with the number of the regions in the network and the number of the processes in its region, respectively, and an input set  $N$  with the identities of its neighbors.

Consider a network where the processes are partitioned into  $m$  regions. Each region has  $n$  processes. Each node in this network can be uniquely identified by two identifiers  $i$  and  $j$ . Identifier  $i$  indicates the region to which the process belongs. Identifier  $j$  indicates the process in region  $i$ . Thus, the processes in the network constitute a process array with two identifiers as follows.

**process**  $p[i: 0..m-1, j: 0..n-1]$

In this network, when a data message is to be sent, the index (bps, eps, br, er, dr, bpd, epd, dp, sr) is attached to the message before the message is sent. When a data (bps, eps, br, er, dr, bpd, epd, dp, sr) message arrives at a process, the process uses its routing table and the index (bps, eps, br, er, dr, bpd, epd, dp, sr) to determine the preferred neighbor to which the message is forwarded.

The routing table of each process  $p[i, j]$  consists of two small tables named *rgn* and *prs*. Table *rgn* determines the preferred neighbor for a destination process whose region is other than  $i$ . Table *prs* determines the preferred neighbor for a destination process whose region is  $i$ , but the process is other than  $p[i, j]$  itself.

The two tables are declared in process  $p[i, j]$  as follows.

**variables**  
     *rgn*           : **array**[ $0..m-1$ ] of  $N$ ,  
     *prs*           : **array**[ $0..n-1$ ] of  $N$

In this declaration,  $N$  is an input set of index ( $i', j'$ ) such that  $p[i', j']$  is a neighbor of  $p[i, j]$ . Each process  $p[i, j]$  has several variables. Variables *interrgn* and *interprs* store candidates for intermediate regions and nodes, respectively. That is, *interrgn* stores the region id's of regions which are at most  $k$  hops away from region  $i$ . *interprs* stores the process id's of local processes which are at most  $k$  hops away from process  $p[i, j]$ . Variable *rgn*[ $w$ ] stores the preferred neighboring region to reach destination region  $w$ . Variable *prs*[ $x$ ] stores the preferred neighboring process to reach destination process  $x$ . Variable *hoprgn*[ $w$ ] stores the distance to reach region  $w$ . Variable *hopprs*[ $x$ ] stores the distance to reach local process  $x$ .

In HLBR, at least one process in each region is called a gateway of the region. The way a process  $p[i, j]$  routes a data(bps, eps, br, er, dr, bpd, epd, dp, sr) message depends on whether  $p[i, j]$  is a gateway of its own region  $i$ . When a data(bps, eps, br, er, dr, bpd, epd, dp, sr) message is received by process  $p[i, j]$ , the process checks to see if the current process is a gateway. Then, if the process is a gateway, the process checks fields *dr*, *dp*, *bps*, *br* and *bpd* of the message. If  $dr = i$  and  $dp = j$ , the message has reached its destination node and is delivered. If  $dr = i$ ,  $dp \neq j$  and  $bpd = 0$ , then the message is in its first routing path in the destination region, and it is routed towards process  $p[prs[epd]]$ . If  $dr = i$ ,  $dp \neq j$  and  $bpd = 1$ , then the message is in its second routing path in the destination region, and it is routed towards process  $p[prs[dp]]$ . If  $dr \neq i$ ,  $er \neq i$  and  $br = 0$ , then the message is in its first global routing path in the network, and it is routed towards process  $p[rgn[er]]$ . If  $dr \neq i$ ,  $er = i$  and  $br = 0$ , then the process sets *br* to 1 and the message starts to be in its second global routing path in the network, and it is routed towards process  $p[rgn[dr]]$ . If  $dr \neq i$ ,  $er \neq i$  and  $br = 1$ , then the message is in its second global routing path in the network, and it is routed towards process  $p[rgn[dr]]$ .

If the process is not a gateway, the process checks fields *dr*, *dp*, *bps*, *br* and *bpd* of the message. If  $dr = i$  and  $dp = j$ , the message has reached its destination node and is delivered. If  $dr = i$ ,  $dp \neq j$  and  $bps = 0$ , then the message is in its first routing path in the destination region, and it is routed towards process  $p[prs[epd]]$ . If  $dr = i$ ,  $dp \neq j$

and  $bps = 1$ , then the message is in its second routing path in the destination region, and it is routed towards process  $p[prs[dp]]$ . If  $dr \neq i$ ,  $eps \neq j$  and  $bps = 0$ , then the message is in its first local routing path in the source region, and it is routed towards process  $p[prs[eps]]$ . If  $dr \neq i$ ,  $eps = j$  and  $bps = 0$ , then the process sets  $bps$  to 1 and the message starts to be in its second local routing path in the source region, and it is routed towards process  $p[prs[gtwy]]$ . If  $dr \neq i$  and  $bps = 1$ , then the message is in its second local routing path in the source region, and it is routed towards process  $p[prs[gtwy]]$ .

Process  $p[i, j]$  in the hierarchical load-balanced routing protocol can be defined as follows.

```

process p[i: 0..m-1,                { i is the region of the process }
                j: 0..n-1]          { j is the process }

constants
  m: integer,      { Number of regions in the network }
  n: integer,      { Number of processes in the region i }

inputs
  N: set of { [i' j'] | p[i', j'] is a neighbor of p[i, j] },
  Z: set of { v | v is a process in the region i },
  gtwy: 0..n-1      { gateway in the local region i }

variables
  interrgn: set of { 0..m-1 }, { a set of candidates for intermediate regions }
  interprs: set of { 0..n-1 }, { a set of candidates for intermediate nodes }
  rgn: array [0..m-1] of N,      { global routing table }
  prs: array [0..n-1] of N,      { local routing table }
  hoprgn: array [0..m-1] of 0..m, { distance to each region }
  hopprs: array [0..n-1] of 0..n, { distance to each local node }
  sr: 0..m-1,                    { source region }
  dr: 0..m-1,                    { destination region }
  dp: 0..n-1,                    { destination node }
  eps: 0..n-1,                   { intermediate node in source region }
  er: 0..m-1,                    { intermediate region }
  epd: 0..n-1,                   { intermediate node in destination region }
  w: 0..m-1,                     { auxiliary }
  x: 0..n-1,                     { auxiliary }
  bps: 0 .. 1, { bps=0:1st routing phase, bps=1:2nd routing phase to the
  gateway in the source region }
  br: 0 .. 1, { br=0:1st routing phase, br=1:2nd routing phase to the dest. region }
  bpd: 0 .. 1, { bpd=0:1st routing phase, bpd=1:2nd routing phase to the
  dest. node in the dest. region }
  kr: integer, { maximum length to intermediate region of 1st routing
  phase in the network }
  kp: integer, { maximum length to intermediate node of 1st routing
  phase in the region i }
  v: element of N

begin
true  $\rightarrow$ 
  kp := average{ hopprs[x] | 0  $\leq$  x < n  $\wedge$  0  $\leq$  hopprs[x] < n }
  interprs := { x | hopprs[x]  $\leq$  kp }
[] gtwy = i  $\rightarrow$ 
  kr := average{ hoprgn[x] | 0  $\leq$  w < m  $\wedge$  0  $\leq$  hoprgn[w] < m }
  interrgn := { w | hoprgn[w]  $\leq$  kr }
[] true  $\rightarrow$  { create and route a new data message to any destination }
  br, bps, bpd := 0, 0, 0;
  dr, dp := any, any; { destination }
  sr := i;
  eps := random(interprs); { chooses an intermediate node in source region }
  RTMSG
[] rcv data(bps, eps, br, er, dr, bpd, epd, dp, sr) from any p[v]  $\rightarrow$ 
{ receives a data message }
  if gtwy = j  $\wedge$  dr = i  $\rightarrow$  { arrived at the gateway in the dest. region }
    epd := random(interprs) { chooses an intermediate node in the
  dest. region }
  [] gtwy = j  $\wedge$  sr = i  $\rightarrow$  { arrived at the gateway in the source region }
    er := random(interrgn) { choose an intermediate region in the
  entire network }
  fi;
  RTMSG
[] gtwy = j  $\rightarrow$  { broadcast the global link status of the current gateway }

```

```

for each p(v)
  send updrgn(hoprgn) to p[v]
rof
[] true  $\rightarrow$  { broadcast local link status }
for each p(v)
  send updprs(hopprs) to p[v]
rof
[] rcv updrgn(hrgn) from any p[v]  $\rightarrow$  { receives global link status }
  if gtwy = j  $\rightarrow$  { update the global routing table }
    UPDRGN
  fi
[] rcv updprs(hprs) from any p[v]  $\rightarrow$  UPDPRS { receives local link status }

```

Statement RTMSG is defined as follows.

```

if gtwy = j  $\rightarrow$ 
  if dr = i  $\rightarrow$  { arrived at destination region }
    if dp = j  $\rightarrow$  skip { arrived, deliver message }
    [] dp  $\neq$  j  $\wedge$  epd  $\neq$  j  $\wedge$  (hopprs(dp)=n  $\vee$  hopprs(epd)=n)  $\rightarrow$  skip
    { destination not reachable }
    [] dp  $\neq$  j  $\wedge$  (hopprs(dp) < n  $\vee$  hopprs(epd) < n)  $\rightarrow$  { destination is reachable }
      if bpd = 0  $\wedge$  epd  $\neq$  j  $\rightarrow$  { still in 1st local routing phase in destination region }
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[epd]]
      [] bpd = 0  $\wedge$  epd = j  $\rightarrow$  { arrived at intermediate node, initiate 2nd
  local routing phase in destination region }
        bpd := 1;
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[dp]]
      [] bpd = 1  $\rightarrow$  { still in 2nd local routing phase in destination region }
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[dp]]
    fi;
  fi;
[] dr  $\neq$  i  $\wedge$  er  $\neq$  i  $\wedge$  (hoprgn(dr)=m  $\vee$  hoprgn(er)=m)  $\rightarrow$  skip { destination
  region is not reachable }
[] dr  $\neq$  i  $\wedge$  (hoprgn(dr) < m  $\vee$  hoprgn(er) < m)  $\rightarrow$  { destination region is reachable }
  if br = 0  $\wedge$  er  $\neq$  i  $\rightarrow$  { still in 1st global routing phase }
    send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[rgn[er]]
  [] br = 0  $\wedge$  er = i  $\rightarrow$  { arrived at intermediate region, initiate 2nd
  global routing phase }
    br := 1
    send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[rgn[dr]]
  [] br = 1  $\rightarrow$  { still in 2nd global routing phase }
    send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[rgn[dr]]
  fi;
fi;
[] gtwy  $\neq$  j  $\rightarrow$ 
  if dr = i  $\rightarrow$  { arrived at destination region }
    if dp = j  $\rightarrow$  skip { arrived, deliver message }
    [] dp  $\neq$  j  $\wedge$  epd  $\neq$  j  $\wedge$  (hopprs(dp)=n  $\vee$  hopprs(epd)=n)  $\rightarrow$  skip
    { destination not reachable }
    [] dp  $\neq$  j  $\wedge$  (hopprs(dp) < n  $\vee$  hopprs(epd) < n)  $\rightarrow$  { destination is reachable }
      if bps = 0  $\wedge$  epd  $\neq$  j  $\rightarrow$  { still in 1st local routing phase }
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[epd]]
      [] bps = 0  $\wedge$  epd = j  $\rightarrow$  { arrived at intermediate node, initiate 2nd
  local routing phase }
        bps := 1;
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[dp]]
      [] bps = 1  $\rightarrow$  { still in 2nd local routing phase }
        send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[dp]]
    fi;
  fi;
[] dr  $\neq$  i  $\rightarrow$ 
  if eps  $\neq$  j  $\wedge$  (hopprs(gtwy) = m  $\vee$  hopprs(eps) = m)  $\rightarrow$  skip
  { the gateway in source region is not reachable }
  [] (hopprs(gtwy) < m  $\vee$  hopprs(eps) < m)  $\rightarrow$  { the gateway in source
  region is reachable }
    if bps = 0  $\wedge$  eps  $\neq$  j  $\rightarrow$  { still in 1st local routing phase in source region }
      send data(bps, eps, br, er, dr, bpd, epd, dp, sr) to p[prs[eps]]
    [] bps = 0  $\wedge$  eps = j  $\rightarrow$  { arrived at intermediate region, initiate 2nd
  local routing phase in source region }
      bps := 1;

```

```

    send data(bps,eps,br,er,dr,bpd,epd,dp,sr) to p[prsgtwy]]
[] bps = 1 → {still in 2nd local routing phase in source region}
    send data(bps,eps,br,er,dr,bpd,epd,dp,sr) to p[prsgtwy]]
fi;
fi;
fi;
fi

```

Statement UPDRGN is defined as follows.

```

hoprgn[i] := 0;
for each w, where w ≠ i do
if rgn[w] = v ^ (hrgn[w]+1) ≠ hoprgn[w] → {if p[i,j] currently routes to region
w through p[v] and p[v]'s distance to region w changes, p[i,j] replaces its table entry }
    hoprgn[w] := min(hrgn[w]+1, m)
[] rgn[w] ≠ v ^ (hrgn[w]+1) < hoprgn[w] → {found a shorter path}
    hoprgn[w] := min(hrgn[w]+1, m);
    rgn[w] := v
[] rgn[w] ≠ v ^ (hrgn[w]+1) >= hoprgn[w] → skip {keep current path}
[] rgn[w] ∉ N → {found an entry in hrgn but not in [i,j]'s table}
    hoprgn[w] := min(hrgn[w]+1, m);
    rgn[w] := v
fi;
rof

```

Statement UPDRPS is defined as follows.

```

hopprs[j] := 0;
for each x, where x ≠ j do
if prs[x] = v ^ (hprs[x]+1) ≠ hopprs[x] → {if p[i,j] currently routes to region x
through p[v] and p[v]'s distance to region x changes, p[i,j] replaces its table entry }
    hopprs[x] := min(hprs[x]+1, n)
[] prs[x] ≠ v ^ (hprs[x]+1) < hopprs[x] → {found a shorter path}
    hopprs[x] := min(hprs[x]+1, n);
    prs[x] := v
[] prs[x] ≠ v ^ (hprs[x]+1) >= hopprs[x] → skip {keep current path}
[] prs[x] ∉ N → {found an entry in hprs but not in [i,j]'s table}
    hopprs[x] := min(hprs[x]+1, n);
    prs[x] := v
fi;
rof

```

## 9. FUTURE WORK AND CONCLUSION

In this paper, we introduced three parameters  $k1$ ,  $k2$  and  $k3$ , in order to exclude the nodes which are too far away from the source from being candidates for the 1<sup>st</sup> intermediate node, from the source gateway from being candidates for the 2<sup>nd</sup> intermediate node and from destination gateway from being candidates for the 3<sup>rd</sup> intermediate node, respectively. One possible direction is to perform simulations with different topologies, different source locations, and different traffic generation distributions, to determine the appropriate values of parameters  $k1$ ,  $k2$  and  $k3$  for improving network performance.

We presented a hierarchical load-balanced routing protocol to distribute the data traffic via bounded randomization over all available paths to a destination in the network for load balancing. The proposed protocol will alleviate congestion and increase the effective bandwidth between each pair of nodes in the network. To accomplish this, each message needs to carry very little extra information: 3 intermediate nodes (e1, e2 and e3) and 3 routing status bits (bps, br, bpd). Moreover each of the 3 intermediate nodes is used at a different time during routing. They may share one space in the message. The required overhead is a space for one node and 3 bits for 3 routing status.

## ACKNOWLEDGMENT

It is our pleasure to thank the anonymous referees for their suggestion.

## REFERENCES

- [1] C. Alaettinoğlu and A. U. Shankar, *The Viewserver Hierarchy for Interdomain Routing: Protocols and Evaluation*, IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, pp. 1396-1410, Oct. 1995.
- [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] L. Breslau and D. Estrin., *Design of Inter-Administrative Domain Routing Protocols*, Proc. ACM SIGCOMM '90, pp. 231-241, Philadelphia, Pennsylvania, Sep. 1990.
- [5] 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 29<sup>th</sup> Annual ACM Symposium on the Theory of Computing, pp. 378-388, May 1998.
- [6] D. P. Bertsekas, *Dynamic behavior of shortest path routing algorithms for communication networks*, IEEE Trans. Automatic Control, AC-27, pp.60-74, 1982.
- [7] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.
- [8] J. A. Cobb and M. G. Gouda, *Balanced Routing*, IEEE Proceedings of the International Conference on Network Protocols, 1997.
- [9] E. W. Dijkstra, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik, Vol. 1, pp. 269- 271, 1959.
- [10] R. C. Dixon, D. A. Pitt, *Addressing, Bridging, and Source Routing (LAN interconnection)*, IEEE Network, Vol. 2, No. 1, Jan.1988.
- [11] J. J. Garcia-Luna-Aceves, *A Minimum-Hop Routing Algorithm Based on Distributed Information*, Computer Networks and ISDN Systems, Vol. 16, pp. 367-382, May 1989.
- [12] M. Gouda, Protocol verification made simple: a tutorial, Computer Networks and ISDN Systems, Vol. 25, pp. 969-980, 1993.
- [13] M. Gouda, *The Elements of Network Protocol Design*, A Wiley-Interscience Publication, John Wiley & Sons, Inc., 1998.
- [14] C. Huitema, *Routing in the Internet*, Prentice Hall, 1995.
- [15] G. Malkin, RIP Version 2, *Internet Request for Comments 1723*, Nov. 1994, Available from <http://www.ietf.cnri.reston.va.us>.
- [16] 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.
- [17] J. Moy, *Ospf Version 2*, Internet Request for Comments 1583, Mar. 1994. Available from <http://www.ietf.cnri.reston.va.us>.
- [18] S. Murthy and J.J. Garcia-Luna-Aceves, *Congestion-Oriented Shortest Multipath Routing*, Proc. IEEE INFOCOM 96, San Francisco, California, Mar. 1996.
- [19] S. Murthy and J.J. Garcia-Luna-Aceves, *Loop-Free Internet Routing using Hierarchical Routing Trees*, Proc. IEEE INFOCOM 97, Vol. 1, pp. 101-108, Apr. 1997.
- [20] T. Nesson and S. L. Johnsson, *ROMM Routing on Mesh and Torus Networks*, Proceedings of the 7<sup>th</sup> Annual ACM Symposium on Parallel Algorithms and Architectures, July 1995.
- [21] C. V. Ramamoorthy and W. Tsai, *An adaptive hierarchical routing algorithm*, Proc. IEEE COMPSAC 83, pp. 93-104, Chicago, 1983.
- [22] Segall and M. Sidi, *A Failsafe Distributed Protocol for Minimum Delay Routing*, IEEE Trans. on Commun., COM-29(5), May 1981.
- [23] D. Sidhu, R. Nair and S. Abdallah, *Finding Disjoint Paths in Networks*, Proceedings of the 1991 ACM SIGCOMM Conference, 1991.
- [24] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.
- [25] Z. Wang and J. Crowcroft, *Shortest Path First with Emergency Exits*, Proceedings of the 1990 ACM SIGCOMM Conference, 1990.
- [26] 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.