Compact Routing in Directed Networks with Stretch Factor of Two

Punit Chandra and Ajay D. Kshemkalyani

Dept. of Computer Science
University of Illinois at Chicago, Chicago, IL 60607-7053, USA.
pchandra@eecs.uic.edu, ajayk@cs.uic.edu

Abstract. This paper presents a compact routing algorithm with stretch less than 3 for directed networks. Although for stretch less than 3, the lower bound for the total routing information in the network is $\Omega(n^2)$ bits, it is still worth examining to determine the best possible saving in space. The routing algorithm uses header size of $4 \log n$ and provides round-trip stretch factor of 2, while bounding the local space by $[n - \sqrt{n}(\sqrt{1 - 7/4n}) - 1/2] \log n$. These results for stretch less than 3 for directed networks match those for undirected networks.

1 Introduction

1.1 Background

As high-speed networking gains popularity, the routing bottleneck shifts from the propagation delay to the route decision function. Therefore, simple routing schemes implemented in hardware will be preferred in practice. The decision function is bounded by the size of the routing table. The bigger the routing table, the more time it takes to determine the next hop. Also, it is desirable that the routing table be kept in fast memory such as cache. Furthermore, it is not desirable for the memory requirements to grow fast with the size of the network, since it means adding more hardware to all the routers in the network.

Compact routing addresses this problem by decreasing the table size and hence the decision time. The trade-off involved here is the communication cost of passing messages between a pair of nodes.

1.2 Existing Work

Early work on compact routing focused on routing schemes for special networks such as rings, trees [14], and grids [9,10]. Peleg and Upfal [11,12] were the first to construct an universal compact routing scheme. An universal routing scheme is an algorithm which generates a routing scheme for every given network, that is, for all sorts of topologies. A trivial version of an universal routing strategy stores at each node a routing table which specifies an output port for each destination. Although this routing strategy can guarantee routing through the shortest path,

B. Monien, V.K. Prasanna, S. Vajapeyam (Eds.): HiPC 2001, LNCS 2228, pp. 24–35, 2001.

each router has to store locally $O(n \log d)$ bits of memory, where d is the degree of the node and n is the total number of nodes in the network. As the network grows in size, it becomes necessary to reduce the amount of memory kept at each router. But there is a trade-off involved here between the memory and the stretch factor, denoted here by s and defined as the maximum ratio between the length of the path traversed by a message and that of the shortest path between its source and destination. The lower bounds for routing information in the network are:

For stretch factor	$s \geq 1$	$\Omega(n^{1+1/(2s+4)})$ bits	[13]
For stretch factor	$s < 3$	$\Omega(n^2)$ bits	[4, 6]
For stretch factor	$s = 1$	$\Theta(n^2 \log n)$ bits (optimal)	[5]

There are various algorithms for compact routing. The algorithm in [1] achives a stretch of 3 while using $O(n^{3/2} \log n)$ bits in total, but some individual nodes use $O(n \log n)$ bits. A recent algorithm of [7] uses interval routing for undirected graphs to achieve a stretch of 5 while using local routing table size of $O(n^{1/2} \log n)$ bits. More recently, [2] presented an hierarchical scheme which uses a local routing table size of $O(n^{2/3} \log^{4/3} n)$ and guarantees a stretch of 3.

Most of the recent work done in routing has been for undirected graphs. Directed networks are much harder than the undirected case because a deviation from the exact shortest route has serious consequences. That is, if we take a walk in the wrong direction for a few steps from u, in an undirected graph it is easier to retrace our steps, but in a directed graph it might be extremely hard to get back to u. So to overcome this difficulty, [3] used round-trip distance. The round-trip distance is a measure of how easy it is to get back from an exploratory trip. They achieved a roundtrip stretch of $2^{l+1} - 1$ using $O(l \log n)$ size addresses and a $O(ln^{1/(l+1)})$ sized routing table on the average on each node, where l is an integer greater than 0 and denotes the level of landmark.

Although designing routing algorithms for stretch of less than 3 does not seem so attractive because of the lower bound of $\Omega(n^2)$ as shown in [4,6], it is still worth examining to determine the best possible saving in space. Recently, [8] addressed the problem of compact routing for stretch factor of less than 3 in undirected networks. It presented an algorithm whose local table size is $(n - \sqrt{n} + 2) \log n$ for a stretch factor of 2. In this paper, we propose a simple routing scheme for directed networks. We use a scheme similar to [8], coupled with the concept of round-trip distance which makes it a harder problem. The algorithm uses node names of size $4 \log n$ and has a round-trip stretch factor of 2, while bounding the local space by $[n - \sqrt{n}(\sqrt{1 - 7/4n}) - 1/2] \log n$.

2 Some Definitions Used in Compact Routing

Definition 1. *Routing scheme (R)*

Routing scheme (R) is a distributed algorithm that consists of distributed data structures in the network and a delivery protocol.

Definition 2. *Round-trip distance*

Round-trip distance between any two nodes, say x and y, in a network is the shortest distance from x to y and back. Formally,

$$RT(x, y) = d(x, y) + d(y, x)$$

where $d(x, y)$ represents the shortest distance between x and y.

Definition 3. *Round-trip distance for a routing scheme (R)*

Round-trip distance for a routing scheme (R) between any two nodes, say x and y, is the distance of the path taken by a packet going from x to y and back, in accordance with the routing algorithm (R). Mathematically,

$$RT(R, x, y) = d(R, x, y) + d(R, y, x)$$

where $d(R, x, y)$ represents the distance of the path taken by a packet going from x to y in accordance with the algorithm.

Definition 4. *Round-trip stretch of a routing scheme (R)*

Round-trip stretch of a routing scheme (R) is defined as follows

$$\text{Round-trip stretch (R)} = max\frac{d(R, x, y) + d(R, y, x)}{d(x, y) + d(y, x)}$$

3 Overview of the Algorithm

The initial step of the algorithm is to construct two sets U and W from the set of the network nodes V such that U and W satisfy the following conditions: $V = U \cup W$, $U \cap W = \phi$ and $|U| = |W| = n/2$. The sets are constructed in an incremental fashion. At each step i, u_i and w_i are added to U and W respectively, where $1 \leq i \leq n/2$ and $u_i, w_i \in V - \{U + W\}$. The pair (u_i, w_i) is chosen such that $RT(u_i, w_i)$, the round-trip distance between u_i and w_i, is minimum among all the pairs in $V - \{U + W\}$.

Now suppose $u_i, u_j \in U$, $w_i \in W$, and $i < j$. Also let $RT(u_i, u_j) = z$, $RT(u_i, w_i) = y$, and $RT(u_j, w_i) = x$. See figure 1.

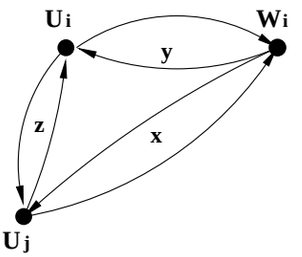

Fig. 1. Round-trip distance for u_i,w_i and u_j

Observation 1. *Given the above construction,* $RT(w_i, u_i) \leq RT(u_j, w_i)$ *and* $RT(w_i, u_i) \leq RT(u_j, u_i)$*, i.e.,* $y \leq x$ *and* $y \leq z$*.*

Proof. The proof follows from the fact that if $RT(u_j, w_i)$ was less than $RT(u_i, w_i)$, then the pair (u_j, w_i) would have been selected before (u_i, w_i). Similar proof applies for (u_j, u_i). □

Note the way the above construction scheme for U and W differs from that used in [8]. The above scheme guarantees $RT(w_i, u_i) \leq RT(u_j, w_i)$ and $RT(w_i, u_i) \leq RT(u_j, u_i)$ while that in [8] guarantees only $d(w_i, u_i) \leq d(u_j, w_i)$. The second condition i.e., $RT(w_i, u_i) \leq RT(u_j, u_i)$, is crucial for the algorithm to work in the case of directed networks.

Consider the case when $z \leq x$. Using Observation 1, we can conclude that $z + y \leq 2x$. In other words, a packet going from u_j to w_i can be routed via u_i if we are ready to tolerate a round-trip stretch of 2. Thus we can omit the entry for w_i in the routing table of u_j. Let us now consider the other case when $x \leq z$. Again by Observation 1, we can conclude that $x + y \leq 2z$. Thus packets from u_j to u_i can be routed via w_i, making it possible to omit u_i in u_j's routing table.

By the above discussion, it is clear that we can always omit $j - 1$ entries from u_j and w_j. The drawback of this scheme is that there is no lower bound on the local space saved. For example at u_1 and w_1, no space is saved. To overcome this, we use the approach used by Cowen [2] which stores the routing information in the header of the packets. For the node u_1, we choose k nodes from the set $V - \{u_1 + w_1\}$, and use headers of the form $H(x_i) = (x_i, y_i, (u_1, P_{(x_i,u_1)}))$, where k is the minimum number of entries we want to save at each node. Here, $H(x_i)$ represents the header of the packet with destination x_i (one of the k nodes), x_i and y_i are the nodes added to U and W respectively in the i^{th} step , and $P_{(x_i,u_1)}$ is the port on u_1 which connects to the first node on the shortest path to x_i. Thus, we need not store an entry for x_i or any of the k nodes at u_1. In a similar manner, we choose k nodes for w_1 from the set $V - \{u_1 + w_1\}$.

In general for u_i, $k - i + 1$ nodes are selected from the set $V - \{u_1 + u_2 + ...u_i + w_1 + w_2 + ...w_i\}$. If x_i is one of the $k - i + 1$ nodes, we set the header of the packet with destination x_i as $H(x_i) = (x_i, y_i, (u_i, P_{(x_i,u_i)}))$. This way we can omit a total of $(i - 1 + k - i + 1) = k$ entries at every node.

Note that for u_i, $k-i+1$ nodes are selected from the set $V-\{u_1+u_2+....u_i+w_1+w_2+.....w_i\}$, which is a dynamic set, i.e., it varies with i, while in [8] the set from which $k-i+1$ nodes were selected is $V-\{u_1+u_2+....u_k+w_1+w_2+.....w_k\}$, which is static. This leads to a more optimized bound on k in case of the above algorithm.

4 The Algorithm

The routing algorithm consists of four parts: an initial construction where nodes are divided and paired according to some set criterion, a header construction where the header is constructed for each destination, a table construction which describes the routing table stored at each node, and the delivery protocol which describes how to identify the output port on which the packet gets sent, based on the header and local routing table.

4.1 Initial Construction

Assume $G = (V, E)$ is a connected directed network with n nodes. Here we construct two equal sets U and W such that $U \cup W = V$, $U \cap W = \phi$ and $|U| = |W| = n/2$.

- *Initiate*: $U = W = \phi$
  For $i = 1$ to $n/2$
  Find (u_i, w_i) such that $RT(u_i, w_i)$ is minimum in $V - \{U + W\}$
  Add u_i to U and w_i to W

Note u_1, w_1 will have the minimum round-trip distance among all the nodes.

4.2 Header Construction

The header for any packet consists of a 3-tuple. The first entry for the header of a packet with destination u_i is u_i itself, the second consists of w_i. The third entry consists of a pair: a node and a port on that node which leads to the shortest path to u_i.

Here we construct a mapping which helps in selecting the third entry in the header. Map u_k to $u_{n/2}$ and u_{k-1} to $\{u_{n/2-1}, u_{n/2-2}\}$ where $k < n$. In general, u_{k-i} is mapped to $\{u_{n/2-\sum_{j=1}^{i} j}, \ldots, u_{n/2-(\sum_{j=1}^{i+1} j)+1}\}$, denoted by $M(u_{k-i})$. Note $|M(u_{k-i})| = |M(w_{k-i})| = i + 1$. For the nodes $M(u_{k-i})$, the third entry in the header is selected as $(u_{k-i}, P_{(x,u_{k-i})})$, where $x \in M(u_{k-i})$. Formally, the $i + 1$ headers for packets whose destination is a node in $M(u_{k-i})$ are

$$H(u_{n/2-\sum_{j=1}^{i} j}) = (u_{n/2-\sum_{j=1}^{i} j}, w_{n/2-\sum_{j=1}^{i} j}, (u_{k-i}, P_{(u_{n/2-\sum_{j=1}^{i} j}, u_{k-i})}))$$

$$\vdots$$

$$H(u_{n/2-(\sum_{j=1}^{i+1} j)+1}) =$$

$$(u_{n/2-(\sum_{j=1}^{i+1} j)+1}, w_{n/2-(\sum_{j=1}^{i+1} j)+1}, (u_{k-i}, P_{(u_{n/2-(\sum_{j=1}^{i+1} j)+1}, u_{k-i})}))$$

The headers for destinations in W are constructed similarly.

4.3 Constructing Tables

The following procedure is used to construct the routing table at node u_j. See figure 2.

- A node, say u_i, is not included in the routing table of u_j if any of the following conditions are satisfied.

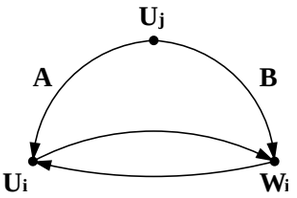

Fig. 2. Table construction for u_j

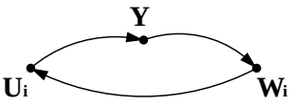

Fig. 3. The route from u_i to w_i

- The node $u_i \in M(u_j)$, i.e., the third element in the header of u_i contains the field u_j.
- $A \geq B$ and $RT(u_i, w_i) < RT(u_j, u_i)$.
- $A \geq B$, $RT(u_i, w_i) = RT(u_j, u_i)$, and either of the two conditions is satisfied:
  - u_j does not lie on the round-trip path from u_i to w_i.
  - u_j lies on the round-trip path from u_i to w_i and $j > i$.

– Else, u_i is included in the routing table of u_j.

Similar procedure holds when deciding about w_i, except that w_i is not omitted from $u_j's$ routing table in the case when $j > i$ and u_i has been omitted from $u_j's$ routing table.

4.4 Delivery Protocol

A packet with header $(u, w, (v, P_{(u,v)})$ at node x is routed according to the following procedure.

– If $x = v$, then route along the port $P_{(u,v)}$.
– Else if u is in $x's$ routing table, then route along the port $P_{(u,x)}$.
– Else if w is in $x's$ routing table, then route along the port $P_{(w,x)}$.
– Else route fails.

5 Correctness Proof

Theorem 1. *The routing algorithm is correct.*

Proof. There are four cases to consider.

Case 5.1. *A packet has to be routed from x $(= u_i)$ to w_i.*
From the table construction scheme, it can be concluded that u_i will have an entry for w_i. Let y $(= u_j)$ be any node between u_i and w_i (Fig. 3). Now we need to show that y has an entry for w_i. This gives five subcases.

1. $d(y, u_i) > d(y, w_i)$: y will have an entry for w_i because this subcase falls in the else part of the table construction scheme.
2. $d(y, u_i) \leq d(y, w_i)$ and $RT(y, w_i) < RT(u_i, w_i)$: again y will include w_i because of the same reason as above.

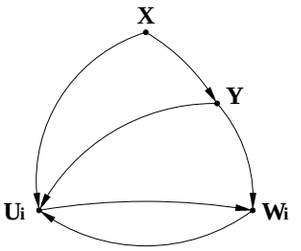

Fig. 4. The route from x to w_i

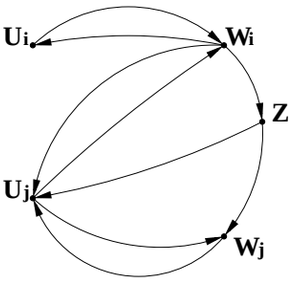

Fig. 5. The route from u_j to w_i and back

3. $d(y, u_i) \leq d(y, w_i)$, $RT(y, w_i) = RT(u_i, w_i)$, and $j < i$: this subcase also falls in the else part of the table construction scheme. Hence, y will have an entry for w_i.
4. $d(y, u_i) \leq d(y, w_i)$, $RT(y, w_i) = RT(u_i, w_i)$, and $j > i$: By using Observation 1, $RT(u_i, w_i) \leq RT(y, u_i)$. Also $RT(u_i, w_i) \geq RT(y, u_i)$, as y is on the round-trip path from u_i to w_i. This gives $RT(u_i, w_i) = RT(y, u_i)$. Expanding the above expression gives

$$d(u_i, y) + d(y, w_i) + d(w_i, u_i) = d(u_i, y) + d(y, u_i)$$

This implies

$$d(y, w_i) + d(w_i, u_i) = d(y, u_i)$$

As $d(w_i, u_i) > 0$, we get $d(y, w_i) < d(y, u_i)$ which contradicts the condition assumed. Hence this subcase cannot exist.
Note that if a packet has to be routed from y to u_i and $j > i$, then the path taken by the packet is via w_i. Also this path is equal to the shortest path from y to u_i as $d(y, w_i) + d(w_i, u_i) = d(y, u_i)$.
5. $d(y, u_i) \leq d(y, w_i)$ and $RT(y, w_i) > RT(u_i, w_i)$: As y is an intermediate node on a round-trip from u_i to w_i, hence $RT(y, w_i) \leq RT(u_i, w_i)$. This is again a contradiction. Thus this subcase cannot exist.

Hence the packet can always be routed from u_i to w_i.

Case 5.2. *A packet has to be routed from x ($\neq u_i$) to w_i and $d(x, u_i) > d(x, w_i)$ (Fig. 4).*
As $d(x, u_i) > d(x, w_i)$, x will have an entry for w_i in its routing table. Let y be any node on the shortest path from x to w_i. Using the triangle inequality, we have $d(x, y) + d(y, u_i) > d(x, u_i)$. As $d(x, u_i) > d(x, y) + d(y, w_i)$, we get

$$d(x, y) + d(y, u_i) > d(x, u_i) > d(x, y) + d(y, w_i)$$

Thus, $d(y, u_i) > d(y, w_i)$. This implies that y will also contain an entry for w_i and the packet will eventually reach w_i.

Case 5.3. *A packet has to be routed from x ($\neq u_i$) to w_i with the following conditions:*

$$d(x, u_i) \leq d(x, w_i) \text{ and } RT(x, w_i) < RT(u_i, w_i)$$

The table construction scheme and the latter condition imply that x will contain an entry for w_i. Now suppose y is any node on the shortest path from x to w_i (Fig 4). Using the fact that $RT(y, w_i) \leq RT(x, w_i)$ and $RT(x, w_i) < RT(u_i, w_i)$, it can be concluded that

$$RT(y, w_i) < RT(u_i, w_i)$$

Hence, y will have an entry for w_i.

Case 5.4. *A packet has to be routed from x ($\neq u_i$) to w_i with the following conditions:*

$$d(x, u_i) \leq d(x, w_i) \text{ and } RT(u_i, w_i) \leq RT(x, w_i)$$

If x is on the round-trip path from u_i to w_i, this reduces to Case 5.1. Otherwise the entry for w_i is omitted from x. Hence the packet moves towards u_i. Let z ($= u_j$) be a node on the shortest path from x to u_i. There are two subcases.

- z does not have an entry for w_i. The packet is routed towards u_i. Once it reaches u_i, this reduces to Case 5.1.
- z has an entry for w_i. This is possible under three scenarios.
  - $d(z, w_i) < d(z, u_i)$
  - $d(z, w_i) \geq d(z, u_i)$ and $RT(z, w_i) < RT(u_i, w_i)$
  - z lies on the round-trip path from u_i to w_i, $j < i$, and $RT(z, w_i) = RT(u_i, w_i)$.

  The first two scenarios represent Case 5.2 and Case 5.3, respectively. The third scenario reduces to Case 5.1. Hence the packet will be routed to w_i via z.

Similar proof holds for destination u_i. □

6 Complexity Analysis

In this section, we find the bounds on maximum space and round-trip stretch.

6.1 Round-Trip Stretch

Theorem 2. *The upper bound on round-trip stretch for the algorithm presented in Section 4 is two.*

Proof. There are four cases to be considered. See figure 5.

Case 1. *A packet has to be routed from x ($= u_i$) to w_i.*

This corresponds to the first case in the previous section. Remember that any node between u_i and w_i will have an entry for w_i and a node between w_i and u_i will have entry for u_i. Thus the round-trip stretch for (u_i, w_i) is 1.

For the rest of the cases, we will consider the round-trip from x ($\neq u_i$) to w_i as the route for the packet.

Case 2. *A packet has to be routed on the round-trip path from x (where $x \neq u_i$ and $x = u_j$) to w_i with the condition that x has an entry for w_i and w_i has an entry for u_j.*

As u_j has an entry for w_i, so any node between u_j and w_i will also have an entry for w_i (Case 5.2.). Similar argument holds for any node between w_i and u_j. Hence the packet will travel through the shortest round-trip path between u_j and w_i. This again gives a round-trip stretch of 1.

Case 3. *A packet has to be routed on the round-trip path from x (where $x \neq u_i$ and $x = u_j$) to w_i and x has an entry for w_i while w_i does not have an entry for u_j.*

The packet will travel from u_j to w_i along the shortest path. From w_i, it will be routed towards w_j. There are two subcases to be considered.

- None of the nodes between w_i and w_j has an entry for u_j. In this subcase, the packet reaches w_j and from there it goes to u_j. This gives

$$RT(R, u_j, w_i) = d(u_j, w_i) + d(w_i, w_j) + d(w_j, u_j)$$

  Note, $d(w_i, w_j) \leq d(w_i, u_j)$ and $RT(u_j, w_j) \leq RT(u_j, w_i)$ because w_i does not have an entry for u_j. Thus,

$$\begin{aligned} RT(R, u_j, w_i) &\leq d(u_j, w_i) + d(w_i, u_j) + d(w_j, u_j) \\ &\leq RT(u_j, w_i) + RT(u_j, w_j) \\ &\leq 2RT(u_j, w_i) \end{aligned}$$

  Hence we obtain a round-trip stretch of 2.
- There is a node between w_i and w_j which has an entry for u_j. Let the first node between w_i and w_j which has entry for u_j be z. The path taken by the packet is from w_i to z and then z to u_j. As $d(w_i, z) + d(z, u_j) \leq d(w_i, w_j) + d(w_j, u_j)$ and $d(w_i, w_j) \leq d(w_i, u_j)$, we get

$$\begin{aligned} RT(R, u_j, w_i) &= d(u_j, w_i) + d(w_i, z) + d(z, u_j) \\ &\leq d(u_j, w_i) + d(w_i, u_j) + d(w_j, u_j) \\ &\leq RT(u_j, w_i) + RT(u_j, w_j) \\ &\leq 2RT(u_j, w_i) \end{aligned}$$

  This also gives a round-trip stretch of 2.

Case 4. *A packet has to be routed on the round-trip path from x (where $x \neq u_i$ and $x = u_j$) to w_i and w_i has an entry for u_j while u_j does not have an entry for w_i.*

Using a similar argument as in the previous case, it can be proved that a round-trip stretch of 2 results.

Note that we cannot have a case where neither u_j has an entry for w_i nor w_i has an entry for u_j. Thus, the round-trip stretch for the algorithm is bounded by 2. □

6.2 Maximum Local Space

Lemma 1. *Node u_i has an entry for either u_j or w_j, where u_i, u_j, $w_j \in V$ and $i > j$.*

Proof. By using Observation 1, it can be concluded that $RT(u_j, w_j) \leq RT(u_j, u_i)$ and $RT(u_j, w_j) \leq RT(w_j, u_i)$. There are 2 cases to be considered.

Case 6.1. *The round-trip path from u_j to w_j contains u_i.*

As u_i is an intermediate node on the round-trip path from u_j to w_j, hence $RT(u_i, w_j) \leq RT(u_j, w_j)$. Thus we can conclude that $RT(u_i, w_j) = RT(u_j, w_j)$. It can be easily verified that for the above condition, either w_j or u_j is included in $u_i's$ routing table.

Case 6.2. *u_i does not lie on the round-trip path from u_j to w_j.*

There are three subcases to be considered.

- $d(u_i, u_j) < d(u_i, w_j)$. From the table construction scheme, it can be concluded that only u_j is included in $u_i's$ routing table.
- $d(u_i, u_j) = d(u_i, w_j)$. Only w_j will be included in $u_i's$ routing table as $RT(u_i, u_j) \geq RT(u_j, w_j)$.
- $d(u_i, u_j) > d(u_i, w_j)$. Again, only w_j will be included in $u_i's$ routing table.

□

Theorem 3. *The maximum local space at any node for the algorithm in Section 4 is $[n - \sqrt{n}(\sqrt{1 - 7/4n}) - 1/2] \log n$.*

Proof. To get the upper bound for maximum local space, consider a node u_i, where $i < k$. Using Lemma 1 and the header construction scheme, we can omit $i - 1 + |M(u_i)| = i - 1 + k - i + 1 = k$ entries at u_i. To bound k, note that

$$\sum_{i=1}^{k} |M(u_i)| \;=\; \sum_{j=1}^{k} j \;\leq\; n/2 - 1$$

which gives $k(k+1) \leq n - 2$. It can be easily verified the $k = (-1 + \sqrt{4n - 7})/2$ satisfies the inequality. Thus the maximum local space at any node is $(n - 1 - k) \log n = [n - \sqrt{n}(\sqrt{1 - 7/4n}) - 1/2] \log n$. □

6.3 Maximum Global Space

From the algorithm, it can be seen that for any node u_i or w_i, where $i \leq k$, k entries are saved. This means that the total space occupied by the routing table on the first k nodes in U and W is $2(n-1-k)k \log n$. For other nodes u_i and w_i, $i-1$ entries are saved on each node. This gives a total table space of $2\sum_{i=n/2}^{n-1-k} i \log n$. Thus the total space for all the nodes is $2((n-k-1)k + \sum_{i=n/2}^{n-1-k} i) \log n$. Substituting the value of k in the above expression gives the total space as $(3n^2/4 - 3n/2 + 2) \log n$.

Note that this scheme saves on an additional $n-2$ entries by using the technique used by Cowen [2] to store the routing information in the packet's header.

7 Conclusion

We presented a compact routing algorithm with stretch less than 3 for directed networks. Although for stretch less then 3, the lower bound for the total routing information on the network is $\Omega(n^2)$ bits, nevertheless we examined it for directed networks to determine the best possible saving in space. The minimum local memory needed for stretch between 2 and 3 is $(n - 2\sqrt{n}) \log n$ for undirected networks [8], while this algorithm requires a maximum local space of $[n - \sqrt{n}(\sqrt{1 - 7/4n}) - 1/2] \log n$ with a stretch of 2 for directed networks.

Acknowledgements. This work was supported by the U.S. National Science Foundation grant CCR-9875617.

References

1. Awerbuch, B., Bar-Noy, A., Linial, N., Peleg, D.: Improved routing strategies with succinct tables. Journal of Algorithms 11 (1990) 307-341
2. Cowen, L. J.: Compact routing with minimum stretch. In Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (1999) 255-260
3. Cowen, L. J., Wagner, C. G.: Compact roundtrip routing in directed networks. In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (2000) 51-59
4. Fraigniaud, P., Gavoille, C.: Memory requirement for universal routing schemes. In Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing (1995) 223-230
5. Gavoille, C., Perennes, S.: Memory requirement for routing in distributed networks. In Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (1996) 125-133
6. Gavoille, C., Gengler, M.: Space-efficiency of routing schemes of stretch factor three. In Proceedings of the 4th International Colloquium on Structural Information and Communication Complexity (1997)
7. Gavoille, C., Peleg, D.: Compact routing scheme with low stretch factor. In Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing (1998) 11-20

8. Iwama, I., Kawachi, A.: Compact routing with stretch factor of less than three. In Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (2000) 337
9. Leeuwen, J. van, Tan, R.: Routing with compact routing tables. In G. Roznenberg and A. Salomaa, editors, The Book of L. Springer-Verlag, New York, New York (1986) 256-273
10. Leeuwen, J. van, Tan, R.: Interval routing. The Computer Journal 30 (1987) 259-273
11. Peleg, D., Upfal, E.: A tradeoff between size and efficiency for routing tables. In Proceedings of the 20th Annual ACM Symposium on Theory of Computing (1988) 43-52
12. Peleg, D., Upfal, E.: A tradeoff between size and efficiency for routing tables. Journal of the ACM 36 (1989) 510-530
13. Peleg, D.: An overview of locality-sensitive distributed computing. Unpublished Monograph, The Weizmann Institute, Rehovot, Israel, 1997
14. Santoro, N., Khatib, R.: Implicit routing in networks. The Computer Science Journal 28 (1985) 5-8