Telecommunication Systems 25:3,4, 287–298, 2004

Proposal for *st*-Routing Protocol

MOHAMMAD MURSALIN AKON, SHAH ASADUZZAMAN and MD. SAIDUR RAHMAN
mm_akon@cs.concordia.ca, asad@cs.mcgill.ca, saidur@nishizeki.ecei.tohoku.ac.jp
Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh

MITSUJI MATSUMOTO mitsuji.matsumoto@ties.itu.ch
Waseda University, Tokyo, Japan

Abstract. A routing protocol chooses one of the several paths (routes) from a source node to a destination node in the computer network, to send a packet of information. In this paper, we propose a new routing protocol, which we call *st*-routing protocol, based on *st*-numbering of a graph. The protocol fits well in noisy environments where robustness of routing using alternative paths is a major issue. The proposed routing protocol provides a systematic way to retry alternative paths without generating any duplicate packets. The protocol works for only those networks that can be represented by biconnected graphs.

Keywords: routing protocol, multi-path routing, biconnected graph, *st*-numbering

1. Introduction

Routing is the way of finding a path from a source to a destination in a network [Keshav, 16]. In other words, routing is the way of deciding which output line will be used to transmit an incoming packet [Tanenbaum, 25]. A router uses a specific routing algorithm to construct a routing table. Routing table consists of entries having a destination and at least the next hop of the probable best path towards that destination, directly or indirectly. Routing algorithms play the role of computing the best path from a source to all possible destinations, depending on some parameters of the underlying network. Way of collecting all the information for the computation described, is one of the concerns of the routing protocol. According to the OSI (Open System Inter-connection) model network layer is responsible to do all the routing jobs. A routing protocol can be designed targeting a particular characteristic of the network or application requirement. But in general, it can be said that a routing algorithm should have the property of correctness, simplicity, robustness, stability, fairness and optimality with minimizing routing table space and control messages, for its acceptance [Mostafa and Singhal, 19]. Most of the requirements seem to be obvious but in practice they often result in contradictory goals. Before going further let us have a look at the options, we have, to design a routing protocol [Maxemchuk and Zarki, 18].

Decision of routing can be taken by a central processor. The responsible central processor then has to maintain status of the whole network, i.e., what are the available links and what are their capacities and among them which are currently up or which

are down, on which links traffic congestion is higher, etc. The processor processes all the information it has and decides the routing table. The constructed routing table is then distributed among the routers. In contrast, all the routers can contribute in creating a stable routing table by a co-operative, distributed protocol. The core telephone networks are of the first kind whereas large computer networks usually represents the later one [Keshav, 16].

Routing can be dynamic (state dependent). State of a network includes network dynamics like congestion of a network or part of a network, delay, cost, etc. Alternative of this type of routing is static (state independent) routing. As static routing does not depend on the current state of the network, the routing table can be constructed considering one or some of the available parameters before the activation of routing operation. Computing such a table is comparatively easy, but the table has to be reconstructed if any of the considered parameters changes in future. Static routing gives better performance for very small networks, whereas state dependent protocols are better for larger networks, though they suffer from network dynamics.

Each router may maintain only one path for any destination. The alternative of maintaining a single path, i.e., maintaining multiple paths requires more table space but gives more freedom in the choice of outgoing path. Beside these, all the routing decision can be made by the source of a packet. The source can include a header with the whole path from the source to destination with each packet. This results in low overhead on other routers on path but the source is needed to be aware of the whole network. If any router along the specified path fails, the packet will be lost. Packet utilization will be lower, if the network is larger, that is if path requires larger space to be described.

In this paper we propose a new routing protocol based on *st*-numberings of graphs. In an *st*-numbering of a graph, each vertex is assigned a number such that at least one adjacent vertex has higher number and at least one adjacent vertex has lower number than the concerned vertex. The source s (numbered as 1) and the sink t (numbered as n for a graph with n vertices) are exceptions. If we obtain an *st*-numbering of a graph, it is easy to find a path from any node to the source or the sink node. In practice, our proposed routing protocol can readily give alternate paths towards any destination using the *st*-numbering scheme.

The rest of the paper is organized as follows. Section 2 describes previous works related to our works. Section 3 deals with graph theoretic terminology and *st*-numbering of graphs. Our proposed routing protocol is given in section 4. Section 5 describes a comparative performance evaluation of the proposed protocol based on simulation models. Finally, section 5 is our conclusion.

2. Related works

Most of the routing protocols found in practice and in literature are of two main genres: distance-vector protocols (e.g., BGP [6], IDRP [Rekhter, 24], RIP [Hedrick, 14; Malken, 17], and EIGRP [Albrightson et al., 1]) and link-state protocols (e.g., ISO IS–IS [15] and OSPF [Moy, 20, 21]). Routers using distance-vector protocols ex-

change path information and compute their preferred path in a distributed manner. The distance-vector protocols are different extensions and implementations of Distributed Bellman–Ford (DBF) algorithm [Bellman, 4] and most of them attempt to eliminate the count-to-infinity problem that prevent the original protocol to converge quickly in highly dynamic networks. The link-state protocols are based on the Dijkstra's distributed shortest path algorithm [Garcia-Luna-Aceves, 11; Dijkstra and Scholten, 9]. Most of these protocols use topology broadcast and incur excessive communication overhead. Several newer variants of link-state protocols (e.g., LVA [Garcia-Luna-Aceves and Behrens, 12] and ALP [Garcia-Luna-Aceves and Spohn, 13]) try to minimize the overhead using dissemination of incremental information only or working with partial topology information. Still most proposals and implementations of both genres are of the type *single path routing* in the sense that they compute a single optimal (or near optimal) path between two nodes at any given state of the network. Since reliability or fault-tolerance is a major design goal of a routing protocol, several new proposals (e.g., DASM [Zaumen and Garcia-Luna-Aceves, 29], MPDA [Vuturuky and Garcia-Luna-Aceves, 26] and MDVA [Vuturuky and Garcia-Luna-Aceves, 27]) compute multiple alternative paths or next hop choices for every destination node. In our proposed solution, we use a vertex numbering scheme, namely, *st*-numbering of a graph that inherently provides this multi-path option.

An *st*-numbering of a graph is well-known in the area of graph theoretical algorithms and plays crucial roles in planarity testing [Booth and Lueker, 5], graph drawing [Battista et al., 8], graph partitioning [Nakano et al., 22], etc. An *st*-numbering of a graph has a beautiful routing property, since each vertex except the source s and the sink t has at least one neighbor with a larger number and at least one neighbor with a smaller number. Recently a mathematical model for network routing is presented in [Annexstein and Berman, 2] based on generalized *st*-numberings. Their main result implies that every 3-connected graph has three spanning trees, which was previously shown in [Zehavi and Itai, 30; Cheriyan and Maheshwari, 7]. However, no attempt to use *st*-numberings in practical network routing is found in literature. To the best of our knowledge, our proposed *st*-routing protocol is the first proposal for using *st*-numberings in practical network routing. Our idea is completely different from that of [Annexstein and Berman, 2]. As we describe in section 4, we compute $n-1$ sets of *st*-numberings of a graph of n vertices considering a vertex as a source and each of the other $n-1$ vertices as a sink, and we prepare routing tables using the computed *st*-numberings. An attractive feature of our proposed *st*-routing protocol is that it can always provide alternate paths in case of congestion or link failure.

3. Preliminaries

In this section we introduce some terminologies on graphs and discuss on *st*-numberings of graphs.

Let $G = (V, E)$ be a connected simple graph with a vertex set V, an edge set E, where $|V| = n$ and $|E| = e$. An edge joining the vertices u and v is denoted by (u, v)

and if (u, v) edge exists we say that u and v are adjacent. The degree of a vertex v is the number of neighbors of v in G.

The connectivity $k(G)$ of a graph G is the minimum number of vertices whose removal results in a disconnected or a single vertex graph. G is called *K-connected* if $k(G) \geqslant K$. If $K = 2$, we call the graph G a *biconnected* graph.

A path $\langle v_0, v_1, \ldots, v_k \rangle$ in a graph G is a sequence of vertices such that (v_{i-1}, v_i) is an edge of G for $1 \leqslant i \leqslant k$. A path is called a u, v-path if the first vertex is u and the last vertex is v. Two paths are called *independent* if they do not share any vertex except their start and end vertices.

The following properties hold for a K-connected (as well as for a biconnected) graph.

Lemma 1 [West, 28]. In a K-connected graph, there exist at least K independent u,v-paths for any pair of vertices u and v in G.

A numbering of the vertices of $G = (V, E)$ by $1, 2, 3, \ldots, n$ is called an *st*-numbering, if two vertices numbered as 1 and n are necessarily adjacent and each j, $1 \neq j \neq n$, numbered vertex is adjacent to two vertices, numbered i and k, such that $i < j < k$. The vertex numbered as 1 is called the *source* and is denoted by s, while the vertex numbered n is called *sink* and is denoted by t.

Not every connected graph has an *st*-numbering, but the following lemma holds.

Lemma 2 [Even, 10; Nishizeki and Chiba, 23]. Let $G = (V, E)$ be a biconnected graph, and let (s, t) be any edge of G. Then G has an *st*-numbering such that s is the source and t is the sink and the numbering can be found in linear time.

One can observe that the following lemma holds for a biconnected graph with *st*-numbering.

Lemma 3. Let $G = (V, E)$ be a biconnected graph with an *st*-numbering. For any vertex $u_1 \neq t$, an u_1, t-path, $u_1, u_2, \ldots, u_k = t$, must exist where u_{i+1} is any neighbor of u_i, $1 \leqslant i \leqslant k - 1$, such that the *st*-number of u_{i+1} is greater than the *st*-number of u_i. Similarly, for any vertex $u_1 \neq s$, an u_1, s-path $u_1, u_2, \ldots, u_k = s$ must exist where u_{i+1} is any neighbor of u_i, $1 \leqslant i \leqslant k - 1$, such that the *st*-number of u_{i+1} is less than the *st*-number of u_i.

In figure 1, an example graph is shown. In the figure the *st*-numbers of different nodes are also given, considering node a as the source and node b as the sink. Now, to traverse from node i to sink node b, we can take the path i, f, e, d, c, b. It can be found that the *st*-numbers on the path are in increasing order. In the similar way, we can take a decreasing ordered *st*-numbered path to reach the source node. Such a path can be i, g, a, having *st*-numbers 5, 4, 1, respectively. For the graph in figure 1, existence of

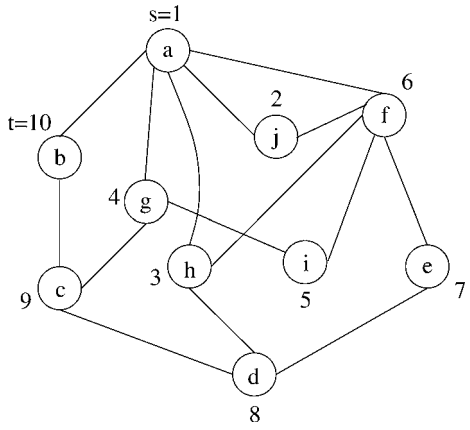

Figure 1. A biconnected graph G with $|V(G)| = 10$ and an *st*-numbering of G taking $s = a$ and $t = b$.

Table 1
Example paths to reach the source (vertex a) and the sink (vertex b) of the graph showed in figure 1.

Vertex	Path to s		Path to t	
	Path	*st*-numbers	Path	*st*-numbers
a	–	–	a, b	1, 10
b	b, a	10, 1	–	–
c	c, g, a	9, 4, 1	c, b	9, 10
d	d, h, a	8, 3, 1	d, c, b	8, 9, 10
e	e, f, a	7, 6, 1	e, d, c, b	7, 8, 9, 10
f	f, a	6, 1	f, e, d, c, b	6, 7, 8, 9, 10
g	g, a	4, 1	g, c, b	4, 9, 10
h	h, a	3, 1	h, d, c, b	3, 8, 9, 10
i	i, g, a	5, 4, 1	i, f, e, d, c, b	5, 6, 7, 8, 9, 10
j	j, a	2, 1	j, f, e, d, c, b	2, 6, 7, 8, 9, 10

such paths for all the vertices are shown in table 1 (vertex sequence and corresponding *st*-number sequence are used to denote the paths).

4. *st*-routing protocol

In this section we describe a new routing protocol which we call the *st-routing* protocol. From the point of view of reliability, we may assume that the underlying graph of our computer network is biconnected.

4.1. Basics of st-routing protocol

Assume that, we have a biconnected graph $G = (V, E)$ with vertex set $V(G)$ and $|V(G)| = n$. For the *st*-routing protocol, we have to find $n - 1$ *st*-numberings of G, taking a particular vertex s as the source and each of the $n - 1$ vertices in $\{V(G) - s\}$

as the sink (referred as t). We denote the *st*-numbering taking the vertex i as the sink as the ith *st*-numbering of G. We will discuss the issue of choosing the vertex s later in this section. There may not always be an (s, t) edge in edge set $E(G)$. In those cases we may assume a dummy (s, t) edge for calculation of *st*-numbering. This does not affect the network topology since we simply omit these dummy edges while the protocol runs.

To run the protocol, each vertex $p \in V(G)$ should maintain the following data structures.

1. A list $STNum(p) = \{STNum(p)_2, STNum(p)_3, \ldots, STNum(p)_n\}$ of integers, $STNum(p)_i$ is the *st*-number of the vertex p in the ith *st*-numbering.
2. An n element table $H(p)$, where ith element is denoted by $H(p)_i$. $H(p)_i$ is a list of adjacent vertices of p such that $q \in H(p)_i$ and $STNum(q)_i > STNum(p)_i$.
3. An n element table $L(p)$, where ith element is denoted by $L(p)_i$. $L(p)_i$ is a list of adjacent vertices of p such that $q \in L(p)_i$ and $STNum(q)_i < STNum(p)_i$.

Seemingly the rows of the table H and L are lists of variable lengths. These two lists are used as the routing table in the *st*-routing protocol. The lists *STNum* are not required for the routing protocol; it is included here for use in the illustrations.

In table 2, example *st*-numbers of different vertices of the graph in figure 1 are shown. For example, *st*-numbers of vertices $a, b, c, d, e, f, g, h, i$ and j vertices are 1, 6, 7, 8, 9, 10, 4, 3, 5 and 2, respectively, when f is the sink vertex. Again from table 2, we can construct $H(f)_q$ and $L(f)_q$. For example, when g is the destination or the sink node, the *st*-number of f ($STNum(f)_g$) is 4 and among the adjacent vertices e and i have higher *st*-numbers (6 and 5, respectively) and a, h, j have lower *st*-numbers (1, 3 and 2, respectively). Hence $H(f)_g = \{e, i\}$ and $L(f)_g = \{a, h, j\}$. The $H(f)$ and $L(f)$ tables for the graph of figure 1 is shown in table 3. Clearly one can construct the tables 2 and 3 in $O(n^2)$ time by lemma 2.

When a router node receives a data packet from another node in the network, it either forwards the packet to one of its neighbors chosen using the routing protocol if

Table 2
st-numbering of the vertices of G in figure 1, where the vertex a is the source vertex and taking different vertices as sinks.

Vertex	Sink vertex								
	b	c	d	e	f	g	h	i	j
a	1	1	1	1	1	1	1	1	1
b	10	9	8	7	6	8	5	7	5
c	9	10	9	8	7	9	6	8	6
d	8	8	10	9	8	7	7	9	7
e	7	7	6	10	9	6	8	6	8
f	6	6	4	5	10	4	9	5	9
g	4	4	7	4	4	10	3	4	3
h	3	3	3	3	3	3	10	3	2
i	5	5	5	6	5	5	4	10	4
j	2	2	2	2	2	2	2	2	10

Table 3
Routing table for the node f in graph G in figure 1.

Destination (q)	$H(f)$	$L(f)$
b	e	a, h, i, j
c	e	a, h, i, j
d	e, i	a, h, j
e	e, i	a, h, j
f	–	a, e, h, i, j
g	e, i	a, h, j
h	h	a, f, i, j
i	e, i	a, h, j
j	j	a, f, h, i

the packet is destined to any other node in the network, or captures the packet if it is destined to the router itself. Assume that a vertex p in the graph has to take a decision about a data packet destined for a vertex q. The following three cases may arise:

(1) $q = p$. Capture the packet.

(2) $q \neq p$ and $q \neq s$. Choose any vertex $v \in H(p)_q$.

(3) $q \neq p$ and $q = s$. Choose any vertex $v \in L(p)_q$.

The proof of correctness of the protocol directly follows from lemma 3. Case 1 is obvious. In case 2, the packet eventually reaches the sink vertex q visiting the vertices of increasing *st*-numbers, since such a path always exists. In case 3, the packet eventually reaches the source vertex s, visiting the vertices with decreasing *st*-numbers.

A few words may be said about the flexibility and robustness of the protocol. When $|H(p)_q| > 1$ for any vertex p and any destination q, the vertex p has more than one vertices to choose from. So, if the chosen link to the neighbor (edge) or the chosen neighbor (vertex) fails, or if the chosen link is congested, p can detect it using an acknowledging mechanism, and then try another vertex $r \in H(p)_q$. If all the vertices of $H(p)_q$ are tried and failed, the packet may be backtracked to a vertex $u \in L(p)_q$, and then try all the vertices in $H(u)_q - \{p\}$. This way all the alternative paths can be tried systematically. Same method works if the destination is s, in which case the vertices of $H(p)_q$ are used for backtracking.

Now, we should find a unique way to choose the source vertex s for the whole graph. We can choose s or source router using any election technique. But in practice we can have a dummy s node, having dummy edges (s, u), for all $u \in V(G)$. In this situation, as we do not have to send any packet to s, for routing packets, it is enough to consider cases 1 and 2 as described above.

In a network with a single *st*-number (say, ith *st*-number), if a packet is generated by the source and each and every node transmits the packet only to all the nodes with higher *st*-numbers, that is nodes in the $H(p)_i$ list for any p and p, $i \in V(G)$, every node of the network will eventually receive the packet. This type of transmission resembles broadcast routing.

4.2. Dynamic st-routing protocol

We may now consider a dynamic scenario of the network, where nodes may come up or may go down from time to time. It is feasible to re-build the routing table from the scratch, as computation for *st*-numbering is cheap. In spite of that, here we propose more simpler protocols to update the routing tables when the network topology changes due to addition of a new node or deletion of an existing node.

4.2.1. Adding a new node (vertex)

First we try to add a vertex. Since the graph should always be biconnected, whenever a new node is added to the network, it should have at least two different adjacent neighbors. Say, a new vertex u is to be added to an n-vertex graph $G = (V, E)$, and the neighbors of u consist of k vertices $a_1, a_2, \ldots, a_k$.

Assuming that the k neighbors $a_1, a_2, \ldots, a_k$ of u have *st*-numbers $n_1, n_2, \ldots, n_k$, respectively, in the ith *st*-numbering ($1 < i \leqslant n$) of the network, and $n_j < n_{j+1}$ for all $1 \leqslant j < k$. n_u can be chosen as the *st*-number of the new vertex u, where $n_1 < n_u \leqslant n_k$. At the first step, vertex u needed to be added to $H(a_1)_i, H(a_2)_i, \ldots, H(a_l)_i$ and $L(a_{l+1})_i, L(a_{l+2})_i, \ldots, L(a_k)_i$ such that n_l is the largest number less than n_u. At the next step, $H(u)_i$ and $L(u)_i$ should be prepared as $H(u)_i = \{a_1, a_2, \ldots, a_l\}$ and $L(u)_i = \{a_{l+1}, a_{l+2}, \ldots, a_k\}$. Finally, all $STNum(p)_i \geqslant n_u$, where $p \in V(G)$, should be incremented by one and $STNum(u)_i = n_u$ should be formally assigned. This last step for updating the *STNum* list is not necessary, if the lists for *STNum* are not physically maintained.

Here the choice of a number n_u between n_1 and n_k as the *st*-number for u does matter. If we choose a number nearer to n_n less number of vertices should change their *st*-numbers, if the *STNum* list is maintained. If a number nearer to n_1 is chosen, we get better number of options for routing without backtracking, because there is a greater number of vertices in the list $H(u)_i$.

All these updates for the tables of ith *st*-numbering should be made for all i from 2 to n. Then a new *st*-numbering taking the vertex u as the sink (the uth *st*-numbering) will be computed and a new row $H(p)_u$ will be added to the table $H(p)$ for every vertex p and a new row $L(p)_u$ will be added to the table $L(p)$ for every vertex p in G. The list $STNum(p)$ for every p will also include a new number $STNum(p)_u$, if the *STNum* list is maintained.

4.2.2. Deleting an existing node (vertex)

Lets u be a vertex of a biconnected graph G and assume that the node representing router u has gone down. According to the assumption, we can still apply *st*-routing protocol, if the graph G continues to be a biconnected graph. In that situation, let v be any of the adjacent vertices of u and $u \in H(v)_i$ for some valid i. We can safely remove u from $H(v)_i$. Problem arises, if $|H(v)_i|$ becomes 0 after the removal of u. In this case v can route packet to $w \in L(v)_i$. We can find such a w, since from lemma 1 we can conclude that if G is a biconnected graph and $|H(v)_i| = 0$, then $|L(v)_i| \leqslant 2$ must

hold. w can detect that the packet is a backtracked packet and can take necessary steps to forward it. The packet will eventually reach the destination, as from lemma 1 there must exist at least another path to node i that does not include v.

Similar procedures can be applied to other adjacent vertices of u. And finally, we can remove the rows $H(q)_u$ and $L(q)_u$ from routing tables of all $q \in V(G)$.

5. Performance analysis

In section 5.1, we describe our simulation model to analyze the performance of our proposed protocol. We also compare the performance of *st*-routing with shortest path routing in section 5.2.

5.1. Simulation model

In our empirical study, we compare the performance of our proposed routing protocol with the static shortest path routing protocol. For the simulation randomly generated graph is taken. To simplify the simulation model, control messages are not considered. In our study, to evaluate the performance of the protocols, we consider total number of packet delivered to the destination as the performance metric.

A custom and event driven simulation tool is built using C++ for performance measurement. Each simulation is run for one hour of simulation clock. Each communication link is considered to have a bandwidth of 1 MB/s. Each router is assumed to have a fixed size buffer to queue incoming packets. The *cost* function, considered for the shortest path routing, is assumed to be depended on several factors of the network: bandwidth, length and data transmission speed of the communication link. It is assumed that cost is proportional to the length of the communication link and inversely proportional to speed and bandwidth.

While choosing next hop, *st*-routing protocol uses a simple heuristic. It chooses the next hop that has the least expected queuing delay. To search such a link only the routers in $H(p)_q$ is considered, where p is current router and q is the final destination, as described in section 4.

5.2. Simulation results

The result of the simulation is presented in figure 2. From the figure, it can be found that the proposed routing protocol is able to deliver more packets in a given amount of time than the all pair shortest path routing protocol. But if the network is too much congested, the performance of the proposed routing protocol degrades. To analyze this characteristic, it is required to note that the proposed protocol does not consider cost of the path. As a result, on an average delivering each packet is costlier than the all pair shortest path protocol. For fair network load, multiple paths help to do load balancing in the network and results in higher throughput. But for higher network load, the increased consumption of resources results in lower throughput.

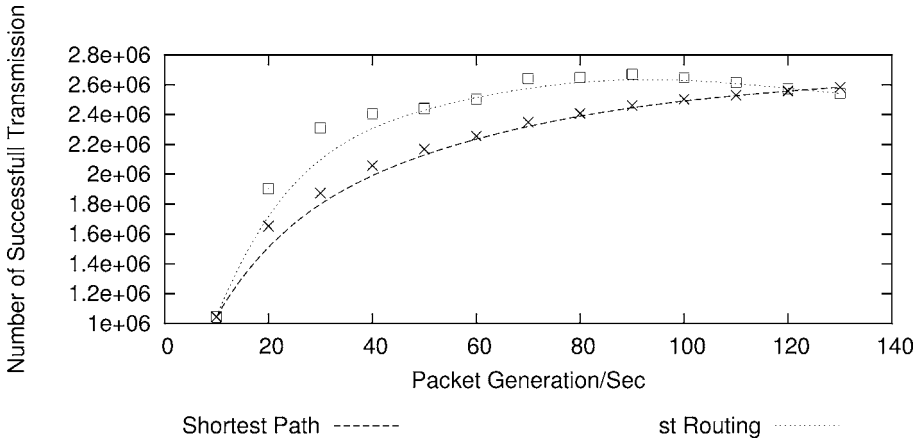

Figure 2. Comparison between shortest path routing and *st*-routing protocols based on throughput.

6. Conclusion

In this paper we proposed a new routing protocol called *st*-routing protocol based on *st*-numberings of graphs. We have presented necessary techniques and required data structures for our proposed protocol.

For a network of n routers, *st*-routing protocol requires memory of $O(nd)$, where d is the maximum degree of the corresponding graph. This is extreme for even some middle range network. This requirement can be reduced by fixing the length of the list $H(q)$ and $L(q)$ for a router q. This will decrease the memory requirement to $O(n)$.

In a network with a single *st*-number, if a packet is generated by the source and each and every node transmits the packet only to all the nodes with higher *st*-number, every node of the network will eventually receive the packet. This type of transmission resembles broadcast routing.

Though some fields for implementation is mentioned, the protocol is not free from limitations. In our *st*-routing protocol we assumed that the network is biconnected from the reliability point of view and the entire topology of the network is known. A distributed algorithm for computing an *st*-numbering of a graph is known which can compute an *st*-numbering of a biconnected graph based on local characterization of a biconnected graph [Aranha and Rangan, 3]. Using that algorithm we are now trying to adapt our protocol in the case where the entire topology is not known.

Acknowledgements

The authors are grateful to the anonymous referees for valuable suggestions which improved quality of the paper.

References

[1] R. Albrightson and J.J. Garcia-Luna-Aceves and J. Boyle, EIGRP – A fast routing protocol based on distance-vectors, in: *Proc. of Networld/Interop '94*, Las Vegas, NV, May 1994.

[2] F.S. Annexstein and K.A. Berman, Directional routing via generalized *st*-numberings, SIAM Journal on Discrete Mathematics 13(2) (2000) 268–279.

[3] R.F.M. Aranha and C.P. Rangan, An efficient distributed algorithm for *st*-numbering the vertices of a biconnected graph, Journal of Universal Computer Science 1(9) (1995) 633–651.

[4] R.E. Bellman, On a routing problem, Quarterly of Applied Mathematics (1958) 87–90.

[5] K.S. Booth and G.S. Lueker, Testing the consecutive ones property, interval graphs, and graph planarity testing using PQ-tree algorithms, Journal of Computer Systems Science 13 (1976) 335–379.

[6] Border gateway protocol 4 (BGP-4), IETF, RFC-1654 (July 1994).

[7] J. Cheriyan and S.N. Maheshwari, Finding non-separating induced cycle and independent spanning trees in 3-connected graphs, Journal of Algorithms 9 (1988) 507–537.

[8] G. Di Battista, P. Eades, R. Tamassia and I.G. Tollis, *Graph Drawing* (Prentice-Hall, Upper Saddle River, NJ, 1999).

[9] E.W. Dijkstra and C.S. Scholten, Termination detection for diffusing computations, Information Processing Letters 11 (1980) 1–4.

[10] S. Even, *Graph Algorithms* (Computer Science Press, Maryland, MD, 1979).

[11] J.J. Garcia-Luna-Aceves, A loop free path finding algorithm: Specification, verification and complexity, in: *INFOCOM*, IEEE, 1995.

[12] J.J. Garcia-Luna-Aceves and J. Behrens, Distributed scalabe routing based on vectors of link states, IEEE Journal on Selected Areas in Communications 13(8) (1995).

[13] J.J. Garcia-Luna-Aceves and M. Spohn, Scalable link-state Internet routing, in: *Proc. of 6th Internat. Conf. on Internet Routing Protocols*, October 1998, pp. 52–61.

[14] C. Hedrick, Routing information protocol, Internet Request for Comment 1058 (June 1988).

[15] Intra-domain IS–IS routing protocol, International Standards Organization, ISO/IEC JTC1/SC6 WG2 N323 (September 1989).

[16] S. Keshav, *An Engineering Approach to Computer Networking* (Addison-Wesley, Don Mills, ON, 1997).

[17] G. Malken, Version 2 – Carrying additional information, Internet Request for Comment 1723 (1994).

[18] N. Maxemchuk and M. El Zarki, Routing and flow control in high speed wide area networks, in: *Proc. of IEEE*, 1990.

[19] W. Mostafa and M. Singhal, A taxonomy of multicast protocols for Internet applications, Computer Communications 20 (1998) 1448–1457.

[20] J. Moy, OSPF version 2, Internet Request for Comment 1247 (1991).

[21] J.T. Moy, *OSPF: Anatomy of an Internet Routing Protocol* (Addison-Wesley, Reading, MA, 1998).

[22] S. Nakano, M.S. Rahman and T. Nishizeki, A linear-time algorithm for four-partitioning four-connected planar graphs, Information Processing Letters 62 (1997) 315–322.

[23] T. Nishizeki and N. Chiba, *Planar Graphs: Theories and Algorithms* (North-Holland, Amsterdam, 1988).

[24] Y. Rekhter, Inter-domain routing protocol (IDRP), Inter-networking: Research and Experience 2(4) (1993) 61–80.

[25] A.S. Tanenbaum, *Computer Networks* (Prentice-Hall, Englewood Cliffs, NJ, 2002).

[26] S. Vuturuky and J.J. Garcia-Luna-Aceves, A simple approximation to minimum delay routing, in: *Proc. of ACM SIGCOMM*, 1999.

[27] S. Vuturuky and J.J. Garcia-Luna-Aceves, MDVA: A distance-vector multipath routing protocol, in: *Proc. of the 20th Annual Joint Conf. of the IEEE Computer and Communication Societies (INFOCOM 2001)*, Vol. 1, April 2001, pp. 557–564.

[28] D.B. West, *Introduction to Graph Theory* (Prentice-Hall, Englewood Cliffs, NJ, 1996).

[29] W.T. Zaumen and J.J. Garcia-Luna-Aceves, Loop free multipath routing using generalized diffusing computation, in: *Proc. of IEEE INFOCOM*, March 1998.
[30] A. Zehavi and A. Itai, Three tree-paths, Journal of Graph Theory 13 (1989) 175–188.