



Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks

SUSANTA DATTA* and IVAN STOJMENOVIC**
SITE, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada

JIE WU

Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL 33431-0991, USA

Abstract. Several localized position based routing algorithms for wireless networks were described recently. In greedy routing algorithm (that has close performance to the shortest path algorithm, if successful), sender or node S currently holding the message m forwards m to one of its neighbors that is the closest to destination. The algorithm fails if S does not have any neighbor that is closer to destination than S . *FACE* algorithm guarantees the delivery of m if the network, modeled by unit graph, is connected. *GFG* algorithm combines greedy and *FACE* algorithms. Greedy algorithm is applied as long as possible, until delivery or a failure. In case of failure, the algorithm switches to *FACE* algorithm until a node closer to destination than last failure node is found, at which point greedy algorithm is applied again. Past traffic does not need to be memorized at nodes. In this paper we further improve the performance of *GFG* algorithm, by reducing its average hop count. First we improve the *FACE* algorithm by adding a sooner-back procedure for earlier escape from *FACE* mode. Then we perform a *shortcut* procedure at each forwarding node S . Node S uses the local information available to calculate as many hops as possible and forwards the packet to the last known hop directly instead of forwarding it to the next hop. The second improvement is based on the concept of dominating sets. Each node in the network is classified as internal or not, based on geographic position of its neighboring nodes. The network of internal nodes defines a connected dominating set, i.e., and each node must be either internal or directly connected to an internal node. In addition, internal nodes are connected. We apply several existing definitions of internal nodes, namely the concepts of intermediate, inter-gateway and gateway nodes. We propose to run *GFG* routing, enhanced by shortcut procedure, on the dominating set, except possibly the first and last hops. The performance of proposed algorithms is measured by comparing its average hop count with hop count of the basic *GFG* algorithm and the benchmark shortest path algorithm, and very significant improvements were obtained for low degree graphs. More precisely, we obtained localized routing algorithm that guarantees delivery and has very low excess in terms of hop count compared to the shortest path algorithm. The experimental data show that the length of additional path (in excess of shortest path length) can be reduced to about half of that of existing *GFG* algorithm.

Keywords: routing, wireless networks, dominating sets

1. Introduction

Wireless networks consist of static or mobile hosts (or nodes) that can communicate with each other over the wireless links without any static network interaction. Each mobile host has the capability to communicate directly with other mobile hosts in its vicinity. They can also forward packets destined for other nodes. Examples of such networks are wireless local area, rooftop, ad hoc and sensor networks. They are used in situations like disaster rescues, wireless conferences in the hall, battlefields, or monitoring objects in a possibly remote or dangerous environment.

The routing problem is the problem of finding a route for sending a message from a source to a given destination. Routing becomes very difficult in wireless networks. In highly mobile situation, the flooding scheme is the most reliable for sending data packets. However, since the link channel re-

source is very scarce and battery power is limited, more efficient schemes must be devised. Numerous routing protocols have been proposed in recent years (see [3]).

Ad hoc networks are best modeled by the *unit* graphs constructed as follows. Two nodes A and B in the network are neighbors (and thus joined by an edge) if the Euclidean distance between their coordinates is at most R , where R is the transmission radius which is equal for all nodes.

The shortest path algorithm does not adapt well to the networks in which some of the nodes may be temporarily inactive due to failures or power savings. In addition to activity status and location updates for all nodes in the network, the updates on the status of every possible link is needed to guarantee the availability of shortest path, which is unacceptable quadratic communication overhead. Wireless networks require *localized* algorithms, in which nodes make routing decisions based solely on the information about its neighboring nodes, and the position of destination. The greedy routing algorithm [5], where each node forwards message to its neighbor that is closest to destination, is based on the location information, which can be supplied even without GPS [4]. Its

* This author is currently with Rational Software, 340 March Road, Ontario K2K 2E4, Canada.

** This author is currently also with DISCA, IIMAS, UNAM, A.P. 20-726, Del. A. Obregon, Mexico D.F. 01000, Mexico.

major drawback is high failure rate for low degree graphs. However, its hop count is very close to the shortest path algorithm, whenever the method is successful. A localized routing algorithm, called *FACE*, that guarantees the message delivery in connected unit graphs, is described [2]. Its improved version, called *GFG* (Greedy–Face–Greedy), that applies greedy algorithm as much as possible, is also described in [2]. We apply the concept of internal nodes to improve the delivery rates of *GFG* algorithm. Another improvement proposed here is a shortcut procedure that allows each node in *FACE* algorithm to find out few next hops in *FACE* algorithm and forward the message directly to the last of these hops.

The desirable properties of any routing protocol include simplicity, loop-free operation, convergence after topological changes, small storage overhead, computational and transmission overhead. The efficiency of a routing algorithm is measured by the delivery rate and the average hop count [15]. The average hop count is the number of transmissions needed by a method between source and destination. It is compared with the average hop count of a shortest path algorithm. The delivery rate is the ratio of numbers of messages received by destination and sent by senders. For a shortest path algorithm, and for algorithms that guarantee delivery in a connected graph, the ratio is 1. Therefore, the only measure used in this paper for comparing routing algorithms that guarantee the delivery is hop count.

The algorithms analyzed and proposed in this paper guarantee delivery if the position of destination is accurate. Sensor networks are important kind of wireless networks whose nodes may be considered fixed. These networks may have thousands of sensors that change their status between active and passive, and therefore shortest path algorithm is not feasible in such case. In some cases nodes in sensor networks may move, but the destination can be a fixed central facility known to all sensors, which collects information from sensors. When the position of destination is fixed, as in this scenario, the algorithms discussed here guarantee delivery as long as the location updates are regularly updated between moving neighboring nodes. If destination is moving, as in the case of ad hoc networks, routing algorithm may begin by sending message to the latest known location of destination, and modify the information by intermediate nodes as the message approaches destination. Since this approach may not guarantee delivery and may increase hop count significantly, we believe that a better approach (e.g. [13]) will apply a destination search method using short probe messages (containing search information but not actual message), then routing from destination to sender with short message, and finally routing full (actual) message from source to destination whose location is detected. The destination search step may apply a flooding with reduced number of retransmissions (such as one based on internal nodes [18]) or another method that is linked to location update scheme (e.g. [13]). In this scenario, delivery is guaranteed unless node movement is significant with respect to message speed (in which case flooding may be the only feasible solution [8]). Moreover, the network can be consid-

ered static in the analysis of algorithm performance, without introducing significant errors.

This paper is organized as follows. Section 2 gives a literature review on localized routing in ad hoc networks, and the concept of dominating sets. Section 3 describes *FACE* and *GFG* algorithms from [2], completed with implementation details and with various enhancements. Section 4 proposes a *shortcut* procedure as the main enhancement to *FACE* algorithm from [2]. Section 5 proposes routing algorithm that restricts routing decisions to internal nodes only. Performance evaluation is given in section 6, followed by conclusion section.

2. Literature review

2.1. Location based routing algorithms

The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors [4]. Alternatively, the location of nodes may be available directly by communicating with a satellite, using GPS, if nodes are equipped with a small low power GPS receiver. We will discuss only location based routing schemes.

Finn [5] proposed a greedy routing algorithm, in which source or any intermediate node will choose the successor node that makes the best progress toward the destination. The distance to destination is used to measure the progress. When no node is closer to the destination than current node C , the algorithm fails. The stoppage criterion differs from the one in *GEDIR* algorithm proposed in [15], where the algorithm stops if the best choice for the current node is to return the message to the previous node. Other existing GPS based routing algorithms are reviewed in [15] and two of them compared experimentally with the *GEDIR* algorithm (other methods were apparently not competitive). One of methods, called *MFR* (most forward with progress) [19] is comparable to *GEDIR* (it chooses the same route as *GEDIR* in over 90% of cases) but is conceptually more sophisticated and requires more power [16], while the other, based on direction of edges [10,11], is not loop-free [15]. Figure 2 gives an example showing that greedy or *GEDIR* algorithm may produce longer paths than *SP* (shortest path) routing algorithm. Power aware routing algorithms in unit graphs are studied in [16]. Routing algorithms that are based on depth first search (DFS) algorithm, merged with *GEDIR*, are proposed in [17]. These algorithms require nodes to memorize past traffic, which is not the case with algorithms discussed here. The performance of *DFS* based routing algorithms can also be improved by applying internal node concept [17].

2.2. Dominating sets and internal nodes

Let G be the set of all nodes in a wireless network, and G' be a subset of G . G' is a dominating set in G if any node in

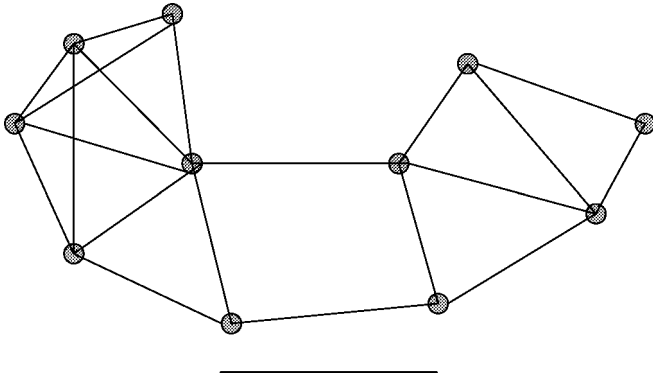


Figure 1. Unit graph and its radius.

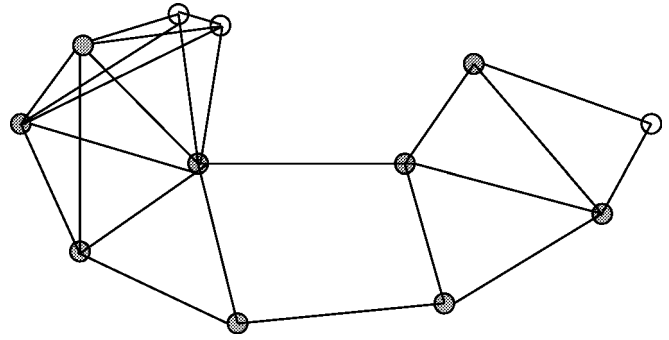


Figure 3. Intermediate nodes of a unit graph.

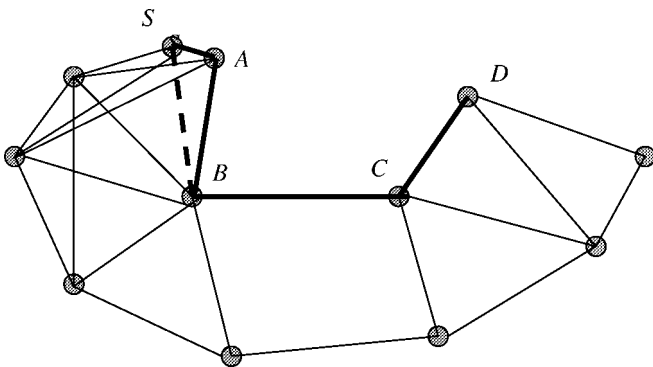
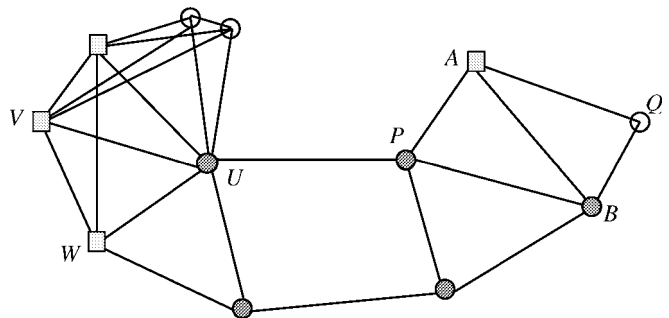
Figure 2. Paths by greedy ($SABCD$) and SP ($SBCD$) algorithms.

Figure 4. Covered intermediate nodes are not gateways.

G either belongs to G' or is neighbor of a node in G' . Nodes that belong to a dominating set G' will be called, in this paper, internal nodes for G (of course, a different definition for dominating set leads to different set of internal nodes). Routing based on a connected dominating set is frequently used approach [21], where the searching space for a route is reduced to corresponding internal nodes. The routing process, in this approach, is divided into three steps. If source node is not an internal node, it forwards the packets to one of its adjacent nodes. This internal node then acts as a new source to route the packets in the reduced graph consisting of internal nodes only. Eventually, the packets reach the destination internal node which is either the destination node itself or neighbor of the destination node. In the later case, the destination internal node forwards the packets directly to the destination node. Such routing is suggested for the shortest path [21], for DSR (dynamic source routing [3]) in [21], and for AODV (Ad Hoc On-Demand Distance Vector) routing in [6], which do not use location information in routing decisions.

Several definitions of internal nodes, their determination by nodes in the network, and application in routing decisions were described in literature. They are mostly based on clustering scheme, which has considerable communication overhead since local changes may trigger global updates (“chain effect” [6]) (see [21] for a more complete discussion and references).

Wu and Li [21] proposed a simple and efficient distributed algorithm for calculating connected dominating set in ad hoc wireless networks. They introduced the concept of an *intermediate* node. A node A is an *intermediate* node if there exist

two neighboring nodes B and C that are not direct neighbors themselves. Consequently, nodes belonging to a shortest path between any two nodes remain intermediate. Figure 3 gives an example of a unit graph, where nine intermediate nodes are marked.

Wu and Li [21] introduced also two rules that considerably reduce the number of internal nodes in the network, and proved that these rules preserve network connectivity after eliminating additional nodes from the dominating set. Let $N(u)$ be the (open) set of all neighbors of node u , and let $N[u] = N(u) \cup \{u\}$ be the corresponding closed neighbor set, that is the set of all neighbors and u itself. Suppose that each node has a unique *id* number (it may be obtained by generating a random number in $[0,1]$, or their x -coordinate may serve the purpose). Let us define *inter-gateway* nodes as intermediate nodes that are not eliminated by Rule 1. Next, let the *gateway* nodes be those intermediate nodes that are not eliminated by both rules. Rule 1 [21] is as follows.

Consider two intermediate nodes v and u . If $N[v] \subseteq N[u]$ in G and $id(v) < id(u)$, then node v is not an inter-gateway node. In other words, if any neighbor of v is also a neighbor of u , and v is connected to u and has lower *id*, then any path via v can be replaced by a path via u , thus node v is not needed as internal node. We may also say that node v is “covered” by node u .

For example, in figure 4, nodes V and W are covered by node U , while node A is covered by node B . In particular, path PAQ via node A can be replaced by path PBQ via node B . We have used x -coordinates of nodes as their *ids*. Thus there are five inter-gateway nodes which are marked in figure 4, while the remaining four intermediate nodes, drawn as small

squares, are not inter-gateway nodes. Observe that the hop count between any two nodes does not increase by applying Rule 1, since a segment pvq of a path between source and destination is replaced by a segment puq .

The number of internal nodes (that is, gateways) can be further reduced by applying Rule 2 [21], as follows. Assume that, after applying Rule 1, u and w are two gateway neighbors of a gateway node v . If $N(v) \subseteq N(u) \cup N(w)$ in G and $id(v) = \min\{id(v), id(u), id(w)\}$, then node v is declared a non-gateway node. In other words, if each neighbor of v is a neighbor of u or w , where u and w are two connected neighbors of v , then v can be eliminated from the list of gateway nodes (when, in addition, v has lowest id among the three). The hop count between a source and destination node may increase by one in this process, since a segment pvq of the path between them is replaced by a segment $puwq$ which is one hop longer. This rule does not eliminate any of the five inter-gateways in figure 4, and thus all of them are also gateway nodes.

Observe that, for both rules, each node may decide to withdraw from the list of internal nodes by knowing its 1-hop neighbors only, since location information (if available) suffices to check whether all of them are neighbors of neighboring internal nodes. Otherwise, list of 2-hop neighbors is necessary for that decision. Thus each node can determine whether or not it is an intermediate node without any message exchanged with its neighbors for that purpose. In order to decide which of neighbors are in dominating set, each node needs to know either position of 2-hop neighbors, or 3-hop neighbors information, or, alternatively, each node needs to add just one bit (referring to its dominating status) in any message announcing its location to all its neighbors.

Stojmenovic, Seddigh and Zunic [18] proposed to replace node ids with a record $(degree, x, y)$, where $degree$ is the number of neighbors of a node, and x and y are its two coordinates in the plane (if available, otherwise they may use a random number instead). In both rules from [21], nodes compare first their degrees, and node with higher degree has greater chances of remaining an internal node. In case of ties, x -coordinate is used to resolve. If x -coordinates also happen to be the same, use y -coordinate for final decision. It is expected that such comparison will result in fewer remaining nodes in the graph, and greater chances of success for a routing method. The example in figure 4 has used so defined record instead of ids . The information about the degree of neighboring nodes may be gathered together with information about their location.

3. Protocol details and modifications to *FACE* and *GFG* algorithms

Bose et al. [2] described a GPS based localized routing algorithm which guarantees the delivery for wireless networks modeled by unit graphs, assuming only that the graph is connected. The only additional constraint is, of course, that the position of destination (as recorded by the sender node) is reasonably accurate. Let G be the graph that corresponds to the

wireless network. Thus it contains n vertices, and two vertices are connected by an edge if and only if they can directly communicate. Since G is unit graph, there exists radius R such that two vertices are connected in G if and only if the distance between them is $< R$.

The algorithm works as follows. In the first phase, construct planar connected subgraph P of G . P is the intersection of unit graph and Gabriel graph GG on the set S of n given nodes, defined as follows. Let $disk(u, v)$ be the disk with diameter (u, v) . Then, the Gabriel graph $GG(S)$ is a graph in which edge (u, v) is present if and only if $disk(u, v)$ contains no other points of S . $GG(S)$ is a planar graph (that is, no two edges cross each other). It was proved in [2] that the intersection of connected unit graph and Gabriel graph (defined over the same set of points) is a connected planar graph. The planar subgraph is constructed in localized manner. More precisely, each node needs to know only the location of its neighbors, and does not need any additional communication step to decide, for each of its edges, whether it belongs to P . Let A be one of nodes, and B one of its neighbors. Edge AB belongs to P if and only if the circle with diameter AB does not contain any other node from G . It suffices to check, for each neighboring node C of A , whether $|CM| < |AB|/2$, where M is the center of the circle, that is the middle point between A and B . Alternative criterion is the following: an edge AB is included in the subgraph if and only if $\angle ACB$ is acute, for each joint neighbor C of A and B . Therefore, the construction takes $O(k)$ computation time for each edge, where k is number of neighbors of A , or $O(k^2)$ time for all edges incident to A . The algorithm is simple, and there exist a more sophisticated construction that takes $O(k \log k)$ computation time (see [2] for reference). An example is given in figure 5, where edges of P are drawn in bold, while thin lines represent edges of G which are not in P .

In the second phase, a path between source node S and destination node D in P is constructed. A higher level description of this phase is given in [2]. We therefore give full details and algorithm here. Planar subgraph P divides the plane into several faces. One of them is open (or infinite, e.g. face $F4$ in figure 3) while other faces are closed (e.g., $F1$, $F2$ and $F3$ in figure 5). The straight line segment between S and D intersects several faces of P , in a particular order ($F1$, $F2$, $F3$, $F4$ and $F5$ in figure 5). Any edge of P belongs to exactly two faces (e.g. AB in figure 5 belongs to $F1$ and $F2$). Suppose that a message m was sent from B to A . If m is always forwarded to a neighbor that has the closest direction to the direction of last traveled edge, in clockwise order, the message will circulate within the same face (in counterclockwise order for a closed face, in clockwise order for an open face). For instance, in figure 5, S sends m to B . B applies clockwise direction and chooses neighbor A , which in turn chooses S (keeping clockwise direction), and finally S sends m again to B . The resulting face traversal appears to be in counterclockwise order, that is $SBASBA$. . . If, on the other hand, the direction is changed at one of edges (say BA) to the closest direction in counterclockwise order, m will circulate around edges of the other face (e.g. B, A, C on face $F2$ in

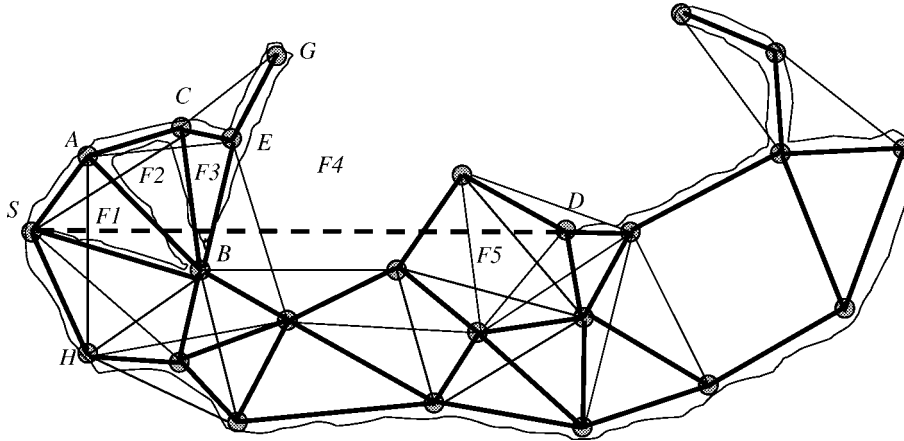


Figure 5. Planar subgraph and path $SBACBEGECASH \dots D$ selected by FACE algorithm.

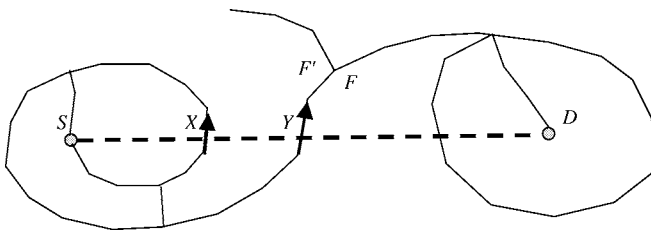


Figure 6. Traversed face does not change in this example.

figure 5), and will never exit that face. Thus, in order to move from one face to the other (e.g., from $F1$ to $F2$), the message needs to change the direction from counterclockwise to clockwise (or *vice versa*). The change shall occur at the edges that intersect the line segment SD (i.e., BA , CB , BE , in figure 3), and each new intersection should be closer to D than previous one. For instance, in figure 5, faces $F1$, $F2$, $F3$ and $F4$ are traversed by applying clockwise, counterclockwise, clockwise and counterclockwise direction at node currently holding m , respectively. The path composed by FACE algorithm is therefore $SBACBEGECASH$ and 14 more hops until D is reached, which happened to be on face $F4$). The path is indicated by the scribble line in figure 5.

The change of faces is not always as regular as in given example. In particular, nonconvex faces may intersect line SD several times, with points of intersection not following any sorted order. The criterion for changing the face is the following. Let X be the point of closest intersection of an edge with SD (initially $X = S$). Suppose that node B sent m to node A and that line segment BA intersects line segment XD at an interior point Y (intersections with SD which are not inside XD are ignored). Let F and F' be the two faces that share AB , and let F be one of them that contains line segment YD . The direction (clockwise or counterclockwise) for further message passing is determined so that m traverses the face F (and $X \leftarrow Y$). Since XD intersects the interior of F at X , F must intersect XD at another interior point (closer to D or D itself). This observation suffices to claim that the algorithm guarantees delivery. Figure 6 gives an example where the direction at the intersection Y shall not change (otherwise a loop is created).

Suppose that the angles (or directions) are positive numbers from the interval $[0, 2\pi)$ (in radians). They can be measured in clockwise or counterclockwise direction (i.e., the angle $\pi/2$ in clockwise direction is equal to the angle $3\pi/2$ if measured in counterclockwise direction). Sender node S may initially measure each angle in both clockwise and counterclockwise directions, and consider the smaller one. It chooses a node B which has closest direction to SD to forward the message (for instance, $\angle BSD < \angle ASD < \angle HSD$ in figure 5 and S selects B). S then attaches its coordinates to the message (in addition to coordinates of D that are normally part of the transmitted message), and coordinates of last intersection X (initially $X = S$) of SD with a face. In addition, S decides the initial direction of message travel within face as follows. If the “winning” neighboring node is measured counterclockwise, then the message travel within the face is performed in clockwise fashion, and angles at current node are measured in counterclockwise direction. In this case, S attaches $sign := 1$ to the message. Otherwise, all mentioned directions are opposite, and $sign := -1$ is attached. Thus every intermediate node is able to determine the line segment XD , and the current direction of message travel. Each intermediate node A applies the following decision in transmitting the message further (if A is not the destination itself):

- If D is a neighbor of A then A can deliver the message directly (however, the algorithm works even if this step is not performed!).
- Let B be the node that sent the message to A . Node A checks whether line segment AB intersects the line segment XD . If it does not, the current direction (i.e. the value of variable $sign$) does not change. On the other hand, if AB intersects XD at a point Y , the sign (direction of travel) may (in most cases) or may not change, and the value of X should be updated to Y for future transmissions. The new value of $sign$ is determined as described below. A selects neighboring node C which minimizes $\angle CAB$, where all angles are measured according to $sign$ (that is, in clockwise direction if $sign = -1$ and in counterclockwise otherwise). In other words, C is the first neighbor that is encountered when direction AB is rotated around A in se-

Function *best-neighbor*($A, B, sign$);
 (* finds the neighbor in planar subgraph with closest direction according to sign *)

```

For each neighbor  $C$  of  $A$  do {
   $Best-direction := 2\pi$ ; (* in radians *)
   $Best-neighbor := B$ ;
   $ACinP := true$ ;
  For each neighbor  $E$  of  $A$  different from  $B$  do
    If  $|EM| < |CA|/2$  then  $ACinP := false$ ; (*  $M$  is middle point of  $AC$  *)
  If  $ACinP = true$  then {
    Find  $\angle CAB$  using usual counterclockwise notion;
    If  $sign = -1$  then  $\angle CAB = 2\pi - \angle CAB$ ;
    If  $\angle CAB < Best-direction$ 
    then  $Best-direction := \angle CAB$ ;  $best-neighbor := C$ };

```

Algorithm *FACE* (S, D); (* initiated by sender node S *)

```

 $A := best-neighbor(S, D, 1)$ ;
 $B := best-neighbor(S, D, -1)$ ;
If  $\min(\angle ASD, 2\pi - \angle ASD) < \min(\angle BSD, 2\pi - \angle BSD)$  then  $B := A$ ;
 $X := S$ ;
If  $D$  is a neighbor of  $S$  then  $S$  sends message to  $D$ 
  Else  $S$  sends message containing  $X, D, sign$  and actual information to  $B$ ;
For each node  $A$  that receives the message from its neighbor  $B$  do { (* initiated by  $A$  *)
  If  $A = D$  then message delivered else
  If  $D$  is neighbor of  $A$ 
  then send message containing  $X, D, sign$  and actual information to  $D$ 
  else {
    If  $BA$  intersects  $XD$  at point  $Y$  then
       $\{X := Y; (x, y) = BA, n = (-y, x), \text{If } n \cdot BD < 0 \text{ then } sign := 1 \text{ else } sign := -1\}$ ;
     $A$  sends message containing  $X, D, sign$  and
    actual information to  $best-neighbor(A, B, sign)$  }

```

Figure 7. Algorithm *FACE*.

lected direction. Note that, in both cases, A may select B to forward the message, and this does not create loop, since B will select a different neighbor in the next turn! Further, loops may appear in the paths, but they are only temporary, since repeated copies of the same message are coming from different neighbors and forwarded to different neighbors.

- The new value of $sign$ is determined as follows. If D is located to the right of vector BA then $sign = 1$ and direction is counterclockwise). Otherwise, $sign = -1$, and direction is clockwise. Whether D is left or right of BA may be decided as follows. Let $BA = (x, y)$ be coordinates of vector BA , and $n = (-y, x)$ be a normal vector to BA (the angle between BA and n is $\pi/2$, measured in counterclockwise notion). D is to the right of BA if and only if dot product $n \cdot BD$ is negative.

The algorithm can be summarized as is shown in figure 7.

The paths generated by *FACE* algorithms are considerably long. However, nodes do not need to memorize past traffic in order to forward the message correctly. Also, there is no need for multiple paths. That is, at each step there is exactly one copy of the message in the network. The *FACE* algorithm may be used in practice in combination with an algorithm that performs well whenever it does not fail, such as greedy routing scheme. Bose et al. [2] proposed an algorithm that begins routing with greedy scheme until message is delivered, or fails at a node C . The failure node is defined as node C which has no closer neighbor to D than itself. Such choice node enables to prove that the combined algorithm remains

loop-free, and guarantees delivery. At C , *FACE* algorithm is invoked which guarantees delivery. The algorithm is further improved, by proposing *GFG* (Greedy-FACE-Greedy) routing scheme. In this scheme, routing follows greedy method until a node C is reached that has no neighbor closer to destination than itself. The distance $d = |CD|$ is attached to the message, and C invokes *FACE* algorithm which runs until message is delivered, or a node B such that $|BD| < d$ is found. Routing then continues again with greedy until message delivery of the next node C that has no closer neighbor to destination. Thus the number of switches from greedy to *FACE* and back may be arbitrary, but each switching node is closer to destination than the previous one. The performance evaluation in [2] confirms that *GFG* method is very practical, and in case of low degree graphs with few hundred nodes it generates paths which are on average up to 3.5 times longer than paths generated by shortest path (*SP*) algorithm. In case of high degree networks, the performance of *GFG* is very close to the performance of *SP* algorithm. This paper attempts to improving the algorithm performance for low degree graphs.

4. Enhancing *FACE* algorithm by shortcut procedure

One of main problems with *FACE* algorithm, causing its increased hop count, is that the planar subgraph construction phase favors short edges over long ones. A route between source and destination is therefore likely to contain a number of short edges. This means that a node A on that route (e.g., $ABCEF \dots$ in figure 6) might have, in addition to forwarding

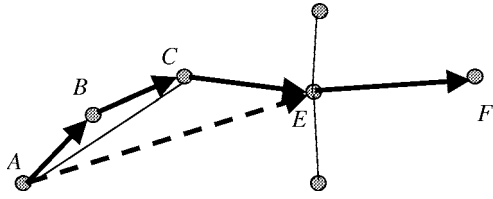


Figure 8. Path $ABCE$ replaced by AE in the shortcut procedure.

node (e.g., B in figure 6), few more nodes in the same route as its neighbors. Can A predict few hops in the route? Consider an example in figure 8, where node A is neighbor of nodes B , C and E on a route $ABCEF \dots$ toward destination. Each node makes forwarding decision based on the location of its neighbors, destination, and the last intersection X of an edge with direction SD (source-destination). Thus node A needs to know the location of each neighbor of B , C and E , in order to predict that the message will traverse indicated route. Instead of forwarding the message to B , A can forward the message directly to F , making a shortcut. Note that A can predict next hops even if face changes.

In order to apply the shortcut procedure, neighboring nodes need to exchange the list and location of their neighbors, in addition to their own location, whenever a change in local topology occurs. In other words, 2-hop local information is needed.

5. Routing via internal nodes

We propose to apply GFG algorithm on the set of internal nodes. The message is sent from source node to one of its neighboring internal nodes that is closest to destination (node that such internal node does not need to be closer to destination than the source itself). The routing algorithm is then restricted to the set of internal nodes. The route follows GFG routing algorithm, non-internal nodes being ignored. The algorithm terminates whenever an internal node that is connected to destination D is reached. At this point, the message is delivered to D . Note that GFG algorithm may be replaced by another routing algorithm, such as $GEDIR$, 2-hop $GEDIR$, flooding $GEDIR$ (these algorithms are proposed and described in [15]), compass routing [10] or MFR [19]. Internal nodes may be defined in one of three mentioned ways.

GFG algorithm, as defined, runs greedy algorithm until a node C is reached that has no closer neighbor than itself to destination D . Let d be distance from C to D . At C , algorithm converts to the $FACE$ algorithm, which has long path but guarantees delivery. When $FACE$ algorithm reaches a node A whose distance to D is $< d$, algorithm switches back to greedy algorithm at A . This conversion may occur 0, 1, 2, 3... times, until message is delivered. One possible improvement for GFG can be defined as follows. We shall define GFG -sooner-back algorithm that runs $FACE$ algorithm until a node B is found that has at least one neighbor that has distance $< d$ to destination. The algorithm switches back to greedy algorithm at node B . This GFG -sooner-back version is likely to return back to greedy algorithm from $FACE$

mode few hops sooner than GFG will do. If B has few neighbors at distance $< d$ from destination, the closest one is chosen, according to greedy scheme. In the performance evaluation given in the next section, GFG refers, in fact, to GFG -sooner-back version. Further improvement of GFG algorithm is obtained by applying shortcut procedure, described in the previous section as GFG - S algorithm. The performance evaluation given in the next section separates the two versions of algorithm, GFG and GFG - S .

6. Performance evaluation

The experiments were carried using random unit graphs, which were generated as follows. Each of n nodes is chosen by selecting its x and y coordinates at random in the interval $[0, 100)$. In order to control the average node degree d , we sort all $n(n-1)/2$ (potential) edges in the network by their length, in increasing order. The radius R that corresponds to chosen value of d is equal to the length of $(nd/2)$ th edge in the sorted order. Dijkstra's shortest path algorithm (from one source to all nodes) can be used to test whether a graph is connected (it is connected if the first node in the network can reach all other nodes by the algorithm). Generated graphs, which are not connected, are ignored. This simple method of generating unit random graphs offers full randomness; however it is the difficulty to generate such graphs for small values of d .

The percentage of internal nodes in a graph is measured in [21] for the lowest ID and in [18] for the highest degree definitions. The data in [18] are found in the context of rebroadcast savings for the flooding task. For $n = 100$ nodes, and degrees in range $d = 4-10$, the percentage of intermediate nodes is 80-95% (increases with d), which does not seem to be significant reduction in size of dominating set. The percentage of inter-gateway nodes, on the other hand, slowly increases from 65 to 70% for same degrees. The best results are obtained for the gateway node definition, where the percentage decreases from 60 to 45%. While these data are significant for broadcasting task, their impact on routing is not so straightforward.

We have tested first the performance of $GEDIR$ algorithm [15] on the internal node set. Surprisingly, no improvements were obtained. It seems that the new algorithm would fail equally frequently but for different source-destination pairs, with no significant improvement in hop count. We then attempted a version where the traversed face is changed before edge crossing with XD , instead of doing it after crossing. There were improvements in the $FACE$ algorithm alone, but it performed worse for GFG algorithm, so we abandoned that approach. Then we made improvements in the basic GFG algorithm, namely inclusion of sooner-back procedure, and comparison with GFG - S version, on the whole set and by applying each internal node definition.

Comparison of intermediate and inter-gateway node based routings show that intermediate node definition performs notably worse but not by large difference. The $FACE$ algorithm alone has up to 8% longer paths, while GFG and GFG - S algorithms have up to 3% longer paths. The differences are larger

Table 1
Hop counts for considered routing methods for $n = 60$ and $d = 3-10$.

	Degree							
	3	4	5	6	7	8	9	10
FACE	33.0	26.4	15.9	12.1	10.7	9.5	7.8	7.6
FACE-S	26.1	18.5	10.3	7.1	5.8	4.9	4.0	3.6
FACE-IG	23.3	18.8	13.0	10.2	9.3	8.3	7.2	6.9
FACE-IG-S	19.5	14.2	8.8	6.4	5.5	4.6	3.8	3.5
GFG	20.5	14.4	7.6	5.2	4.2	3.6	3.1	2.9
GFG-S	17.8	11.7	6.5	4.7	3.9	3.5	3.1	2.9
GFG-IG	15.3	11.5	6.8	4.9	4.1	3.6	3.1	2.9
GFG-IG-S	14.2	10.2	6.1	4.5	3.8	3.4	3.1	2.8
SP	8.9	6.4	5.0	4.1	3.6	3.3	3.0	2.8

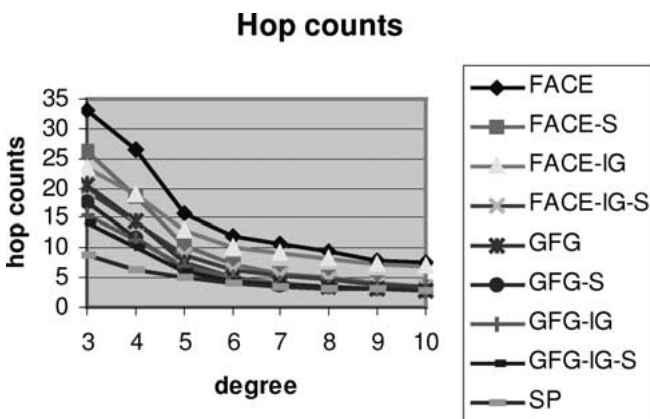


Diagram 1. Hop counts for considered routing methods for $n = 60$ and $d = 3-10$.

for smaller degree graphs. A very close performance of GFG and GFG-S on the inter-gateway and gateway node dominating sets was observed, with differences being always below 1%. Although gateway node definition apparently reduced the size of dominating set, it increases the path lengths at the same time, and, interestingly, the same result at the end is obtained. Because of simplicity and preserving path lengths, we conclude that inter-gateway node definition is the one that has the best performance.

Table 1 and diagram 1 give average hop counts for several routing methods that guaranty message delivery. Averages are taken over 20 connected graphs and each source-destination pairs. Thus for different values of d different graphs were considered. The number of nodes ranges from $n = 40$ to 100, and degrees are 3–10. Data for only inter-gateway node definition of dominating sets are shown. Four such methods are indicated with suffix “IG”. Four methods that apply shortcut procedure have suffix “S”. For $d = 3$ connectivity was provided by requesting next generated node to be connected to the one of previously selected nodes. Such good connectedness for $d = 3$ has also made differences between localized and SP algorithm smaller than for the case $d = 4$. Thus we shall refer to $d = 4$ as being the degree with largest difference in algorithm performance, using SP as benchmark.

Let us define the *dilation* as the ratio of hop count for given method and the hop count of SP (shortest path) algorithm. It

Table 2
Dilations of GFG-IG-S algorithm.

n/d	Degree							
	3	4	5	6	7	8	9	10
40	1.45	1.26	1.15	1.08	1.03	1.02	1.01	1.01
50	1.52	1.43	1.25	1.08	1.04	1.04	1.04	1.02
60	1.60	1.58	1.24	1.10	1.06	1.04	1.01	1.01
70	1.85	1.53	1.26	1.11	1.12	1.02	1.02	1.02
80	2.00	1.74	1.26	1.18	1.10	1.07	1.04	1.02
90	2.17	1.69	1.41	1.16	1.10	1.08	1.04	1.02
100	2.27	2.08	1.41	1.17	1.11	1.04	1.03	1.03

is desirable to have the ratio as close to 1 as possible. The largest dilation reported in [2] for GFG algorithm (without shortcut procedure) was 3.5 (for $n = 100$ and $d = 4$), and our goal was to reduce that number as much as possible. For $n = 60$, the largest dilation reported in [2] was 2.3, for $d = 4$ (no values were reported for $d = 3$).

It can be observed that FACE-S algorithm, that is FACE algorithm enhanced with shortcut procedure, has very low dilation for high degree graphs (under 1.7 for $d \geq 7$). FACE-IG-S (the previous algorithm applied on inter-gateway nodes) comes even closer, under 1.6 for $d \geq 5$. GFG-S reduces the dilation from 2.3 to under 1.9 for $d \geq 4$, GFG-IG to under 1.8, and finally GFG-IG-S to under 1.6. Thus we obtained localized routing algorithm with guaranteed delivery whose excess in hop count with respect to the shortest path algorithm is under 60% for $d \geq 3$ (the results for $d = 3$ were also within the bound). Thus improvement of adding internal node concept and shortcut procedure over former GFG algorithm in terms of dilation is about 30% for lower degree graphs. In fact, the improvement is much higher if measured in terms of excess path rather than full path. Obviously we cannot improve the length of the shortest path, thus we can improve only the length of additional path. In this respect, the 1.3 “added dilation” of GFG [2] is reduced to 0.6 added dilation, thus over 53% reduction.

Table 2 shows the dilations obtained in our experiments for number of nodes ranging from 40 to 100, and for degrees between 3 and 10. Clearly, the proposed localized algorithm compares very well with the global shortest path algorithm, especially for dense networks.

GFG algorithm [2] was implemented (one year later) by Karp and Kung [9] by including MAC layer considerations and experiments with moving nodes. Their GPSR algorithm [9] uses sparser relative neighborhood graph instead of its supergraph, Gabriel graph, used in [2], and changes faces before edge crossings instead of doing it afterwards; both changes make the algorithm worse than previously published GFG algorithm [2]. The experiments [9] confirmed the validity and efficiency of the algorithm [2] under realistic scenarios with moving nodes and message collisions. Since we improve upon GFG algorithm, similar improvements (to the results reported in [9]) may be expected for the case of moving nodes. Handling mobility is a very difficult problem, and the performance will depend on the selected location update scheme. Location update schemes are surveyed

in [13]. Karp and Kung [9] implemented *GFG* algorithm [2] by adding a full IEEE 802.11 MAC protocol, and by experimenting with mobile nodes moving according to a random waypoint model, using ns-2 environment. They used the same simulation code as the measurement study used to evaluate *DSR* protocol [3]. Accurate destination information was used in [9], thus location updates and corresponding overhead were not implemented or measured. Their *GPSR* (Greedy Perimeter Stateless Routing) algorithm [9], extended from *GFG* [2], consistently delivered over 94% (mobility may introduce disconnection) data packets successfully; it is competitive with *DSR* in this respect on 50 node networks at all pause times, and increasingly more successful than *DSR* as the number of nodes increases. The routing protocol traffic generated by *GPSR* was constant as mobility increased, while *DSR* must query longer routes with longer diameter and do so more often as mobility increases (with less effective caching). Thus *DSR* generates drastically more routing protocol traffic in simulations [9] with over 100 nodes. Therefore the scalability seems to be the major advantage of *GFG* algorithm over existing well known protocols.

7. Conclusions

The performance evaluation of our algorithm, as demonstrated in the previous section, demonstrated that very significant improvements were obtained, and that proposed improvements to routing algorithm lead to algorithm that perform very close to the performance of the shortest path algorithm for all graph average degrees.

We propose to study further the reduction in the number of internal nodes. For example, some graphs of regular nature (for instance, square, triangular and hexagonal networks, composed from regular squares, triangles and hexagons, respectively) may declare all nodes as intermediate ones. One approach that can be studied is as follows. Internal nodes make a connected subgraph G^1 of the initial network G . Let G^i be the connected subgraph of internal nodes of the graph G^{i-1} , for $i = 2, 3, \dots, j$, where j is the smallest number such that $G^j = G^{j-1}$. The nodes in the final graph G^j may be considered as a kind of kernel of the initial network G . The kernel may consist of one or two nodes, or maybe a convex polygon, or may have some other shape. A node in the kernel may be elected as a leader by some criteria (e.g., the leftmost node in a GPS based system). It is an interesting idea to study the use of such “hierarchical” graph construction and node division for efficient routing, broadcasting or reduction operations. Reduction operation is a reverse broadcasting, in which information from all nodes is collected at one node that requested it, possibly in a processed form (for example, the sum of all data, one per node).

We developed power aware routing algorithms with guaranteed delivery [14], by merging algorithms from [16] with improved *FACE* algorithm. One of the problems associated with dominating set definition used in this paper is their fixed status in the absence of node mobility, causing more energy

consumption at internal nodes. We are investigating [20] an alternate definition of dominating set, that will take node remaining battery level into consideration. This will enable change of internal node status in favor of nodes with more battery power.

The *GFG* algorithm, enhanced by shortcut procedure in *FACE* mode, and applied on internal nodes, may be applied to solve geocasting problem. In geocasting problem, a message is to be sent from a node to all nodes within a region (e.g., circle). *GFG* may be used to route toward the center of the geocasting region until a point inside the region is reached. The algorithm then switches to flooding with retransmission savings [18].

It is also an interesting problem to find another kind of planar subgraphs which is denser than Gabriel graph (GG). *FACE* routing algorithm may perform better if more faces and edges are present in the subgraph. The new kind of subgraphs needs also to be defined locally, without any message exchange. Delaunay triangulation (DT), for instance, is denser subgraph but is not locally defined. GG is subgraph of DT. It is well known [1] that GEDIR guarantees delivery in DTs, thus DT is a good candidate for study. More precisely, to include the largest possible portion of DT in the planar subgraph, which can be locally found. For instance, each triangle in DT contains no other point from the set, which can be used to include all such triangles whose all sides have size comparable to unit graph radius and therefore locally detectable. Note that the circumscribed circle of that triangle should also be of locally manageable size. Further improvements in the algorithm performance could be obtained by following this approach.

Acknowledgements

The authors are grateful to referees for their constructive comments. This work was supported in part by REDII-CONACYT, NSERC and NSF.

References

- [1] P. Bose and P. Morin, Online routing in triangulations, in: *Proc. of the 10th Annual Int. Symposium on Algorithms and Computation ISAAC* (1999) pp. 113–122.
- [2] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, in: *3rd Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, 20 August 1999, pp. 48–55; *ACM/Kluwer Wireless Networks* 7(6) (November 2001) 609–616.
- [3] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: *Proc. MOBICOM* (1998) pp. 85–97.
- [4] S. Capkun, M. Hamdi and J.P. Hubaux, GPS-free positioning in mobile ad-hoc networks, in: *Proc. Hawaii Int. Conference on System Sciences*, January 2001; *Cluster Computing* 5(2) (2002) 157–167.
- [5] G.G. Finn, Routing and addressing problems in large metropolitan-scale internetworks, ISI Research Report ISU/RR-87-180 (March 1987).

- [6] M. Gerla, T.J. Kwon and G. Pei, On demand routing in large ad hoc networks with passive clustering, in: *Proc. IEEE WCNC* (September 2000).
- [7] S. Giordano, Mobile ad hoc networks, in: *Wireless Networks and Mobile Computing Handbook*, ed. I. Stojmenovic (Wiley, 2002), to appear.
- [8] C. Ho, K. Obraczka, G. Tsudik and K. Viswanath, Flooding for reliable multicast in multi-hop ad hoc networks, in: *Proc. MOBICOM* (1999) pp. 64–71.
- [9] B. Karp and H.T. Kung, GPSR: Greedy perimeter stateless routing for wireless networks, in: *Proc. MOBICOM* (August 2000) pp. 243–254.
- [10] E. Kranakis, H. Singh and J. Urrutia, Compass routing on geometric networks, in: *Proc. 11th Canadian Conference on Computational Geometry*, Vancouver, August 1999.
- [11] Y.B. Ko and N.H. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: *MOBICOM* (1998) pp. 66–75.
- [12] M. Seddigh, J. Solano Gonzalez and I. Stojmenovic, RNG and internal node based broadcasting algorithms for wireless one-to-one networks, *ACM Mobile Computing and Communications Review* 5(2) (2001) 37–44.
- [13] I. Stojmenovic, Location updates for efficient routing in ad hoc networks, in: *Handbook on Wireless Networks and Mobile Computing* (Wiley, 2002), to appear.
- [14] I. Stojmenovic and S. Datta, Power aware routing with guaranteed delivery in wireless networks.
- [15] I. Stojmenovic and X. Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks, *IEEE Transactions on Parallel and Distributed Systems* 12(10) (2001) 1023–1032.
- [16] I. Stojmenovic and X. Lin, Power aware localized routing in wireless networks, *IEEE Transactions on Parallel and Distributed Systems* 12(11) (2001) 1122–1133.
- [17] I. Stojmenovic, M. Russell and B. Vukojevic, Depth first search and location based localized routing and QoS routing in wireless networks, in: *IEEE International Conference on Parallel Processing*, Toronto, 21–24 August 2000, pp. 173–180.
- [18] I. Stojmenovic, M. Seddigh and J. Zunic, Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks, *IEEE Transactions on Parallel and Distributed Systems*, to appear.
- [19] H. Takagi and L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio terminals, *IEEE Transactions on Communications* 32(3) (1984) 246–257.
- [20] J. Wu, M. Gao and I. Stojmenovic, On calculating power-aware connected dominating sets for efficient routing and broadcasting in wireless networks, in: *IEEE Int. Conference on Parallel Processing*, Valencia, Spain, September 2001, pp. 346–353.
- [21] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, *Telecommunications Systems* 18(1–3) (September 2001) 13–36.



Ivan Stojmenovic received the B.S. and M.S. degrees in 1979 and 1983, respectively, from the University of Novi Sad and Ph.D. degree in mathematics in 1985 from the University of Zagreb. He earned a third degree prize at International Mathematics Olympiad for high school students in 1976. In 1980 he joined the Institute of Mathematics, University of Novi Sad (Yugoslavia). He was a visiting researcher at Electrotechnical Laboratory (Tsukuba, Japan) in 1985/6, Washington State University (Pullman, USA) in 1987, and University of Miami (Miami, USA) in 1988. In Fall 1988, he joined the faculty in the Computer Science Department at the University of Ottawa (Canada), where currently he holds the position of a Full Professor in SITE. Since June 2000 he is frequently in Mexico City as researcher in DISCA, IIMAS, UNAM. He published over 150 different papers in journals and conferences, and edited “Wireless Networks and Mobile Computing Handbook” (Wiley, 2002). His research interests are wireless networks, parallel computing, multiple-valued logic, evolutionary computing, neural networks, combinatorial algorithms, computational geometry, graph theory and computer science education. He is currently a managing editor of *Multiple-Valued Logic*, an International journal, and an editor of the following journals: *Parallel Processing Letters*, *IASTED International Journal of Parallel and Distributed Systems*, *Parallel Algorithms and Applications*, and *Tangentia*.
E-mail: ivan@site.uottawa.ca



Jie Wu a Professor and the Director of CSE graduate programs at Department of Computer Science, Florida Atlantic University. He has published over 100 papers in various journal and conference proceedings. His research interests are in the area of mobile computing, routing protocols, fault-tolerant computing, interconnection networks, Petri net applications, and software engineering. Dr. Wu served as a program vice chair for 2000 International Conference on Parallel Processing (ICPP) and a program vice chair for 2001 IEEE International Conference on Distributed Computing Systems (ICDCS). He is a program co-chair of the 12th ISCA International Conference on Parallel and Distributed Computing Systems in 1999. He is also a co-guest-editor of a special issue in *IEEE Transactions on Parallel and Distributed Systems* on “Challenges in Designing Fault-Tolerant Routing in Networks” and a co-guest-editor of a special issue in *Journal of Parallel and Distributed Computing* on “Routing in Computer and Communication Networks”. He is the author of the text “Distributed System Design” published by the CRC press. Dr. Wu a recipient of the 1996–1997 Researcher of the Year Award at Florida Atlantic University. He is also a recipient of the 1998 Outstanding Achievements Award from IASTED. He is an IEEE Computer Society Distinguished Visitor. Dr. Wu is a Member of ACM and ISCA and a Senior Member of IEEE.
E-mail: jie@cse.fau.edu



Susanta Datta received Master Degree in computer science from the University of Ottawa in 2001 and has extensive experience in large-scale software design and development, C/C++ and Java. He is a Software Development Manager in Rational Software.
E-mail: sdatta@rational.com