Telecommun Syst (2010) 44: 159–169
DOI 10.1007/s11235-009-9215-4

Fault-tolerant routing in adversarial mobile ad hoc networks: an efficient route estimation scheme for non-stationary environments

B. John Oommen · Sudip Misra

Published online: 10 November 2009

Abstract Designing routing schemes that would successfully operate in the presence of adversarial environments in Mobile *Ad Hoc* Networks (MANETs) is a challenging issue. In this paper, we discuss fault-tolerant routing schemes operating in a network with malfunctioning nodes. Most existing MANET protocols were postulated considering scenarios where all the mobile nodes in the *ad hoc* network function properly, and in an idealistic manner. However, adversarial environments are common in MANET environments, and misbehaving nodes certainly degrade the performance of these routing protocols. The need for fault tolerant routing protocols was identified to address routing in adversarial environments in the presence of faulty nodes by exploring redundancy-based strategies in networks. It turns out that since the nodes are mobile, the random variables encountered are non-stationary, implying that estimation methods for stationary variables are inadequate. Consequently, in this paper, we present a new fault-tolerant routing scheme that invokes a stochastic learning-based weak estimation procedure to enhance a route estimation phase, which, in turn, is then incorporated in a route selection phase. We are not aware of any reported method that utilizes non-traditional estimates to achieve the ranking of the possible paths. The scheme, which has been rigorously tested by simulation, has been shown to be superior to the existing algorithms.

Keywords Mobile ad hoc networks · Routing · Weak estimation · Learning automata

1 Introduction

MANETs are characterized as an infrastructure-less cooperative engagement of mobile nodes forming networks with continuously changing topologies. They do not have centralized network managers, access points, fixed base stations, or a backbone network for controlling the network management functions and do not possess designated routers for making routing decisions. All nodes in MANETs take part in routing by acting as routers for one another. However, several hops are normally needed in such networks for transmission of data from one node to another because of the limited wireless transmission range associated with the operation of the mobile nodes [2, 7, 9].

The above-mentioned characteristics of MANETs, particularly those arising due to the mobility of nodes, and the continuously changing network topologies, pose several challenges. Due to the continuously changing topologies, the routes that were once considered to be the "best" routes may no longer remain the same at a later time instant. This, therefore, requires a continuous re-computation of routes, and there is no permanent convergence to a fixed set of routes in such networks. So, any routing protocol that

A preliminary version of some of the results here appeared in the *Proceedings of WiMob 2006, the 2006 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Montreal, June, 2006.

B.J. Oommen
School of Computer Science, Carleton University, K1S 5B6, Ottawa, Canada
e-mail: oommen@scs.carleton.ca

B.J. Oommen
Department of Information and Communication Technology, University of Agder, Grooseveien 36, 4876, Grimstad, Norway

S. Misra (✉)
School of Information Technology, Indian Institute of Technology, 721302, Kharagpur, WB, India
e-mail: sudipm@iitkgp.ac.in

needs to operate in MANET environments should take these issues into consideration.

Designing routing protocols poses further challenges when one needs to design routing schemes in the presence of adversarial environments in MANET networks. This is, more precisely, the problem addressed in this paper. Specifically, we discuss fault-tolerant routing schemes where there are malfunctioning nodes in the network. Most existing MANET protocols were postulated considering scenarios where all the mobile nodes in the *ad hoc* network function properly, and in an idealistic manner. However, adversarial environments are common in MANET environments, and misbehaving nodes can, indeed, degrade the performance of these routing protocols [11]. The need for fault tolerant routing protocols was identified to address routing in adversarial environments, specifically in the presence of faulty nodes, by exploring network redundancies [10, 11].

Despite the challenges mentioned above, it is appropriate to highlight a few applications of MANETs which have rendered them popular. One of the popular application domains of MANETs is communication in moving battlefields [7]. Other applications involve communication in rural regions where building up fixed wireline or wireless infrastructures is both costly and difficult.

As discussed, due to the mobility of the nodes and the rapidly changing topologies, the reliability of the correct transmission of messages is an important concern for MANETs. Hence, strategies that would guarantee the delivery of packets in adversarial environments, and in the presence of node/link failures are sought for.

Different routing mechanisms have been proposed in the literature for MANETs—primarily those that work under the assumption of "ideally" behaving environments. Two important schemes that we have used in our study use the *dynamic source routing* (*DSR*) [2, 4, 5], and the *multipath routing* [6, 8, 14] philosophies. Multipath routing protocols discover multiple routes between a pair of source-destination nodes. In multipath routing, multiple redundant packets are sent along different paths between a pair of source-destination nodes. As opposed to this, DSR is a unicast dynamic on-demand routing protocol. It is a source routing protocol, where the source explicitly provides a packet with the complete information of the route to be followed, and this route is used by the intermediary nodes to forward the packet to the right destination node. In DSR, when two hosts need to communicate with each other, the sender host determines a route based on the information stored in its cache, or based on the results of a route discovery, depending on whether or not the information about the destination node is already available to the source node.

The well-known MANET routing algorithms (e.g., DSR, multipath routing) are unsuitable as fault-tolerant routing algorithms for MANETs. Since DSR chooses the shortest path route for packet transmission in adversarial environments, it can be shown that DSR will achieve a low packet delivery rate. On the other hand, multipath routing algorithms are strong in their fault-tolerance ability, because they send multiple copies of packets through all possible (disjoint) routes between a pair of source-destination nodes. However, the disadvantage with multipath routing algorithms is that they introduce an unnecessary amount of overhead on the network. Without a mechanism that "tolerates" route failures due to malfunctioning nodes while making routing decisions, the performance of *ad hoc* network protocols will necessarily be poor, and the routing decisions made by those protocols would be erroneous.

Xue and Nahrstedt [10, 11] confirmed that devising a fault-tolerant routing algorithm for *ad hoc* networks is inherently hard. This is because the problem itself is NP-complete due to the unavailability of correct path information in these environments. Xue and Nahrstedt [10] designed an efficient algorithm, called the *End-to-End Fault Tolerant Routing* (*E2FT*) *Algorithm*, which is capable of significantly lowering the packet overhead, while guaranteeing a certain packet delivery rate.

1.1 Motivation behind our work and our contributions

The spectrum of algorithms that attempt to solve the problem under consideration do so by:

1. Either "flooding" the network with multiple redundant packets along different paths between a pair of source-destination nodes (thus increasing the probability of the successful transfer), or
2. Following a dynamic on-demand routing protocol, where the source *explicitly* provides (*a priori*) the transmitted packet with the complete information of the route to be followed (thus minimizing the number of multiple redundant packets being transmitted), or
3. Seeking a "happy" medium between the latter strategies, namely by estimating the potential profitability of maintaining selected paths.

Our strategy is a combination of all these three philosophies. The rationale for the solution we seek can be catalogued as follows:

1. First of all, we opt to retain *certain* multiple redundant paths, and thus follow the basic principles of the multipath families.
2. Secondly, we simultaneously seek a solution that minimizes the "flooding", thus pursuing the *dynamic* source-routing philosophy.
3. Finally, we seek a solution which is akin to the one proposed in [10, 11], except that we attempt to explicitly consider the nature of the random variables encountered. Observe that since the nodes are *mobile*,

these random variables are, by definition, non-stationary. Thus, rather than use traditional maximum likelihood estimates, we argue that it is expedient to utilize *weak* estimates, namely those that converge in *distribution* as opposed to those that converge *with probability one*. We achieve this by invoking the recently-developed novel *weak* estimation methods, that are built on the principles of stochastic learning—as explained in [12, 13].

To the best of our knowledge, a scheme which collectively uses all of these principles is new to the area of fault-tolerant MANETs. Indeed, more particularly, we are not aware of any reported method which utilizes non-traditional estimates to achieve the ranking of the possible paths. *These are the primary contributions of this paper.*

2 Problem model

The problem model that we have considered is similar to that used by Xue and Nahrstedt [10], with a few differences to simulate more realistic MANET scenarios. Our study, however, incorporates the consideration of non-stationary environments, as discussed later in this section. We consider a graph $G = (V, E)$ consisting of V mobile nodes, and E bi-directional links connecting different nodes. If there are n mobile nodes in a path, the length of any path p is denoted by $L(p)$, in which $p = \{v_1, v_2, \ldots, v_n\}$, where $v_1, v_2, \ldots, v_n \in V$, and $(v_i, v_{i+1}) \in E, i \in \{1, 2, \ldots, n-1\}$. The multipath routes between a pair of source-destination nodes is denoted by $\pi = \{p_1, p_2, \ldots, p_m\}$, where m is the number of paths between any pair of source-destination nodes. In such a model, $L(\pi) = \sum_{i=1}^{m} L(p_i)$ is used to represent the length of the multipath route.

The *packet delivery probability of a path* is represented as $\gamma(p) = \prod_{i=1}^{m} \gamma(v_i)$. If there are m paths in a multipath route between a pair of source-destination nodes, the packet *delivery probability of a multipath route*, $\gamma(\pi)$, determines the probability that when multiple copies of the packets are sent along all the m paths between the source-destination pair, at least one copy is received. $\gamma(\pi)$ is calculated as $\gamma(\pi) = 1 - \prod_{i=1}^{m}(1 - \gamma(p_i))$.

The problem that we address in this paper is that of determining a mechanism for fault-tolerant routing that would route packets through mobile nodes in the above environment (i.e., in the presence of faulty nodes) by providing a certain packet delivery rate guarantee, and at the same time by routing the least number of duplicate packets through multiple routes between a pair of source-destination nodes. The reader should note that "blind" multipath routing algorithms are capable of achieving a high packet delivery rate guarantee, because they utilize the benefits of network redundancy. However, their disadvantage is that they route duplicate packets through the multipath routes to provide such a high packet delivery guarantee. Therefore, we seek a solution that would provide a certain optimum packet delivery rate guarantee, which at the same time would also reduce the "overhead" routing that could burden the network by virtue of the packet duplication mechanisms adapted by the existing "blind" multipath routing algorithms.

Another *objective* of our work is to propose an algorithm that would be efficient in *non-stationary* environments, i.e., environments where the fault probability of a mobile node increases as it moves away from the center of the network in which it is supposed to operate. In other words, as a node moves away from the center of the region of operation, the likelihood of dropping packets also increases. This is an enhancement over the work by Xue and Nahrstedt [10].

3 The E2FT algorithm

In the interest of brevity, our present survey of the E2FT algorithm is necessarily brief. The algorithm involves two major phases: A *route estimation* phase, and a *route selection* phase. The route *estimation* phase is used to estimate the packet delivery probability of all the routes at the disposal of the algorithm at any time instant. As opposed to this, the route *selection* phase is used to select those routes that are confirmed to have satisfied a certain optimization constraint, and drop those routes from further consideration that are estimated to be unnecessary among all the available multipath routes between a pair of source-destination nodes.

In the route estimation phase, the number of packets sent depends on the level of accuracy desired through the estimation process. Note that a superior estimation is achieved by sending a large number of packets, with a tradeoff of the overall high network overhead. The accuracy of the estimation is achieved progressively through iterations.

The route selection algorithm works as follows. At the beginning, since no estimation results are obtained, all paths between a pair of source-destination nodes are selected to route the packets. By using a suitable estimation criterion, when the associated estimates of the paths are guaranteed to be accurate enough, the paths are reviewed to either be confirmed as one of the routes that "wins" the selection process, and be permanently used for routing all future requests, or be dropped from further routing considerations.

4 Proposed solution

As mentioned earlier, the *objective* of our work is to propose an efficient routing algorithm for MANETs, which will be able to minimize the overhead by sending the least possible number of redundant packets, while guaranteeing a certain rate for the delivery of packets. The reader should recall that

there is a tradeoff between the rate of delivery of packets and the overhead. It is possible to achieve a very high packet delivery rate if the number of packets sent is not a concern (e.g., by using the multipath routing scheme). On the other hand, it is possible to achieve a very low overhead if we do not care about the number of packets that are successfully delivered (e.g., by using the DSR scheme). Thus, attempting to increase one, will decrease another, and *vice versa*. What is challenging is how we can achieve a "balance" between the two. In other words, we need an algorithm that will be able to minimize the overhead by guaranteeing a certain level of efficiency of the packet delivery process.

To achieve our objectives, we propose a stochastic learning-based weak estimation fault-tolerant routing scheme.

4.1 Weak estimation learning

In statistical problems involving random variables, the quality, reliability, and accuracy of the estimation are important considerations. Traditionally, there have been different estimation schemes proposed in the literature, which can broadly be classified as either belonging to the *Maximum Likelihood Estimator* (*MLE*) class of algorithms [3, 4], or as belonging to the *Bayesian family* of algorithms [1, 3]. Although the above estimation schemes have been proven to be quite efficient, they work under the premise that the underlying distribution in the environment is stationary, i.e., the estimated parameter does not vary with time. Together with Rueda, the first author studied this problem [12, 13], and proposed a novel estimation scheme for learning in non-stationary environments.[1] They considered the case where the Bernoulli trials yielding binomially distributed outcomes of random variables changed with time to new random values.

In our fault-tolerant routing solution, we have used this efficient procedure for the estimation of the packet delivery probability through available paths. It is called the *Stochastic Learning Weak Estimator* (*SLWE*) scheme[2] [12, 13], and is based on the stochastic learning paradigm. It uses a learning parameter, λ, which does not influence the mean of the final estimate. On the other hand, the variance of the final distribution, and the speed of convergence decrease with the increase in the value of the learning parameter. We now present below the weak estimation scheme.

Let us consider a binomially distributed random variable X, such that:

$$X = \begin{cases} 0 & \text{with probability } s_0 \\ 1 & \text{with probability } s_1 \end{cases} \tag{1}$$

such that $s_0 + s_1 = 1$, where $S = [s_0, s_1]^T$.

At any time t, let X assume the value $x(t)$. In order to estimate s_0 and s_1, WELA keeps track of the running estimate $p_i(t)$ of s_i at time t, where $i = 0, 1$. In such a setting, the value of p_0 is updated using the following *multiplicative* scheme:

$$p_0(t+1) = \begin{cases} \lambda \times p_0(t) & \text{if } x(t) = 1 \\ 1 - \lambda \times p_1(t) & \text{if } x(t) = 0 \end{cases} \tag{2}$$

where λ is a constant ($0 < \lambda < 1$), called the learning parameter, and $p_1(t+1) = 1 - p_0(t+1)$.

We now present below, without proof, some of the interesting results (taken from [12]) concerning WELA.

Theorem 1 *Let X be a binomially distributed random variable, and $P(t)$ be the estimate of S at time 't'. Then, $E[P(\infty)] = S$.*

Theorem 2 *If the components of $P(t+1)$ are obtained from the components of $P(t)$ as per (2), $E[P(t+1)] = M^T E[P(t)]$, where M is a stochastic matrix. Thus the limiting value of the expectation of $P(.)$ converges to S, and the rate of convergence of P to S is fully determined by λ.*

Theorem 3 *Let X be a binomially distributed random variable governed by the distribution S, and $P(t)$ be the estimate of S at time 't', obtained by (2). Then, the algebraic expression for the variance of $P(\infty)$ is fully determined by λ.*

4.2 Proposed algorithm

We use the above-mentioned weak-estimation learning scheme to propose a new fault-tolerant routing algorithm, named *Weak-Estimation-Based Fault Tolerant Routing Algorithm* (*WEFTR*), which is capable of efficiently estimating the probability of delivery of packets through the paths available at any moment. Like the E2FT algorithm [10], the WEFTR algorithm involves, among other steps, a *route estimation* phase, and a *route selection* phase. The route estimation phase is used to estimate the packet delivery probability of all the routes at the disposal at any time instant, whereas the route selection phase is used to select those routes that are confirmed to have satisfied a certain optimization constraint, and drop the unnecessary multipath routes between a pair of source-destination nodes.

[1] The theory of these estimates is presented here, briefly, and without the respective proofs. The details and the proofs can be included if requested by the Referees.

[2] The term "weak" used in the SLWE estimator scheme refers to the weak convergence of the random variable with respect to the first and second moments only.

In the *route estimation* phase, N packets are sent along a path p. The source node estimates the fraction of packets delivered, $\hat{\gamma}(p)$ from the number of packets N' that are received along that path.[3]

In our strategy, the estimate of the packet delivery probability is refined with the increase in the number of iterations. In every iteration, a set of packets is transmitted through each of the multipath routes between a pair of source-destination nodes. We can have two possible scenarios for any path: the nodes in a path either forward the packets correctly, or they do not. Consequently, we can use a binomial estimation scheme (based on the above SLWE) as follows:

$$\hat{\gamma}_0(p) = \begin{cases} \lambda \times \hat{\gamma}_0(p) & \text{if the path does not forward the packet correctly} \\ 1 - \lambda \times \hat{\gamma}_1(p) & \text{if the path forwards the packet correctly} \end{cases} \quad (3)$$

where λ is the learning parameter, such that $0 < \lambda < 1$, and $\hat{\gamma}_1(p) = 1 - \hat{\gamma}_0(p)$.

In our *route selection* algorithm, for a path to be confirmed, the following condition should be satisfied: $\hat{\gamma}_{WE}(p) \geq \gamma^*$, where γ^* is the minimum packet delivery probability required for a path to be confirmed, and $\hat{\gamma}_{WE}(p)$ is the packet delivery probability estimate using the weak-estimation scheme presented in equation (3). Once a path is confirmed, it is considered to be useful for routing future requests, and so no further estimation is carried out on that path.

The dropping algorithm selects a path, $p_{\min}$, from all the available paths, π, with the minimum packet delivery estimation value, where the latter is examined to see if the following dropping condition [10] is satisfied:

$$\hat{\gamma}_{WE^{1/m}}(\pi') \geq \gamma^*$$
$$\text{where } \hat{\gamma}_{WE^{1/m}}(\pi') = 1 - \prod_{p \in \pi'} (1 - \hat{\gamma}_{WE^{1/m}}(p))$$
$$\text{and } \pi' = \pi - \{p_{\min}\} \quad (4)$$

We present below a high level sketch of the proposed algorithm.

Algorithm WEFTR

Input

- A graph (network) with a set of nodes, and a set of links connecting the nodes.
- The nodes are mobile, and links connecting them can be reset with the change in the position of the nodes.
- Some of the nodes in the network are faulty with a certain packet delivery rate dependent on the distance of the node from the center of the area of mobility of the mobile nodes (simulation area).

Output

- All the incoming packets are delivered from the source node to the destination node (with the intention of maximizing the packet delivery rate, and minimizing the network overhead).

Algorithm

BEGIN

Step 0 (initialization) Initialize a vector WEFTR_MP that stores all the paths in use, and WEFTR_Nodes that stores all the nodes in the graph, with information about their estimated packet delivery probability.

At each second, do the following:

Case 1: If the second is a simulation pause then
- Step 1. Save the estimated packet delivery probability of each node in the vector WEFTR_Nodes.
- Step 2. Update the edges and probabilities in the graph to reflect the current position of the nodes and calculate the new paths from the source to the destination.
- Step 3. Use the values stored in WEFTR_Nodes in order to calculate the estimated packet delivery probability of each path.

Case 2: At each second
- Step 4. Try to confirm or drop paths. Paths dropped are removed from the WEFTR_MP vector.
- Step 5. Use all the paths in the WEFTR_MP vector to send the packets, and calculate the number of packets that are received for each path, and the total number of non-duplicated packets that are received.

END

4.3 Experimental details

In order to determine how the performance of the proposed algorithm compares with other competing algorithms,[4] we

[3] Traditionally, this is estimated as: $\hat{\gamma}(p) = \frac{N'}{N}$.

[4] There are currently quite a few algorithms (and their variants) reported in the literature that claim to solve the present problem. It is clearly impossible to compare any single algorithm with *all* of them. But we have opted to compare our new algorithms with individual schemes that represent the various "families" of strategies reported—as described in Sect. 1.1. The rationale for choosing these is that we believe that it represents a reasonably fair comparison against the entire *spectrum* of philosophies *motivating* the algorithms. We are currently doing a more comprehensive comparison (including the testing on "real-life" network topologies).

simulated an *ad hoc* network with mobile nodes, and dynamically changing topologies, and ran our proposed algorithm along with the other benchmark algorithms (described in the next section) in the simulated environment.

4.3.1 Simulation environment

The simulated environment that we considered consists of a flat square of length 500 meters. There are 50 nodes in the network, each having a different data delivery probability which decreases as they move away from the center of the square, and increases as they move closer to it. In other words, if we fix a node in the centre of the square, the reliability of data delivery to its peer nodes and vice versa decreases as those peer nodes move far away from it. This can happen due to diminishing signal strength between any pair of communicating wireless devices when they move away from each other. Furthermore, to assume that things are done in a systematic way as per the benchmark accepted "standards" we assume that each node moves randomly, following the *random waypoint model*.[5] If after a random move as per equations (5) and (6), a node reaches the edge of the square, then the move is canceled and a new random move for the same is done until it lands in a valid position. In our simulated *ad hoc* network, we assumed that the maximum speed with which the mobile nodes can travel is 20 m/s. Observe that the nodes move at each time unit, but the links between them are only recalculated at a simulation pause.[6] The maximum speed of a node specified above (i.e., 20 m/s) is needed to calculate how much a node can move in a second. This is because the position of a node at the ith second is calculated as:

$$X\text{pos}(i) = X\text{pos}(i-1) + \text{randnum} \tag{5}$$

$$Y\text{pos}(i) = Y\text{pos}(i-1) + \text{randnum} \tag{6}$$

In the above, $X\text{pos}(i-1)$ and $Y\text{pos}(i-1)$ denote the abscissa and ordinate of a node in the previous second (or time instant), and $X\text{pos}(i)$ and $Y\text{pos}(i)$ denote the abscissa and ordinate, respectively, of the corresponding node at the current second (or time instant). If the maximum speed is 20 m/s, the random shown above is a random number generated between -20 and $+20$.

[5]The details of this model can be found at http://www.netlab.tkk.fi/~esa/java/rwp/rwp-model.shtml.

[6]Here we assume that the links between the nodes in the network do not get "torn down" with every movement of the nodes in the network. In other words, we assume that the links in the network remain connected until a certain time (i.e., the pause time). The alternative could have been to recompute the links in the network with a unit movement of nodes. The former, according to our view (although debatable), is more realistic. Additionally, recomputing the links with every movement of the nodes in the network would lead to a prohibitively large computational overhead.

The maximum distance two nodes can have for which they are connected (they can deliver packets to each other) is directly dependent on the simulation parameter "Sparsity". The Sparsity of the network is an attribute that signifies how the nodes connect with one another. It denotes a coefficient whose value ranges between 0 and 1: A value of 1 signifies that very few edges (100% sparse coefficient) connect with one another, whereas a value of 0 signifies that the maximum possible number of nodes connect with one another (i.e., a 0% sparse coefficient). The reader should observe that in the simulation there is no fixed number of links in the networks. The links are recalculated at each simulation pause. This is because two nodes are considered to have a link if they are within a certain distance of each other. So the Sparsity directly influences this distance.

Another parameter that is used in the simulations is called the *pause time*. It signifies how the algorithms accommodate to node mobility. This parameter defines the time interval after which the links are recomputed. Each of the simulations was run for 500 seconds. During the simulation period, randomly bit rate (CBR) traffic was generated between a pair of nodes randomly with a rate of 10 KB/s. Also, during the simulations the learning parameter was kept constant, although, as mentioned earlier, the learning parameter does not influence the mean of the final estimate.

We determine how far a node is from the center of the square, by measuring its Euclidean distance from the center.

4.3.2 Benchmark algorithms

In order to establish how our algorithm performs when compared to the existing algorithms, we selected three algorithms, all of which were executed together with our proposed algorithm in the simulated environment. The three algorithms we chose as benchmark algorithms are:

1. DSR Algorithm
2. Multipath Routing Algorithm
3. E2FT Algorithm[7]

Of all these three algorithms, E2FT represents the state-of-the-art in the area of fault tolerant routing in MANETs, and so, we reckoned that the performance comparison between our algorithm, and E2FT is crucial. However, since DSR and the multipath routing algorithms are currently widely used in deployed MANETs, they were also considered. Although DSR is a simple routing algorithm, it is weak when it concerns routing information in the presence of malfunctioning nodes. On the other hand, the multipath routing is, perhaps, a very strong routing algorithm in terms of routing information when there are misbehaving nodes. But, as

[7]In our study, to be fair to the competition, we consider the optimized version of E2FT that provides an optimization methodology—namely one that takes the mobility of the nodes into account.

mentioned earlier, the greatest limitation of multipath routing is that it brings with it a large network overhead, as it loads all relevant routes between a pair of source-destination nodes with redundant packets, so as to ensure that the destination node receives at least one correct copy of the packet sent from the source.

4.3.3 Performance metrics

Two metrics were used for evaluating the performance of the algorithms invoked in the experiments:

1. *Percentage of packets delivered*: This represents the rate of successful delivery of packets to the destination. This is calculated as follows. At each second, the packet delivery probability of all the paths in use is calculated. Then, for each packet sent at that time unit, a random number between 0 and 1 is generated. If the number is lower than the packet delivery probability, the packet is considered as delivered. After all the iterations, the percentage of delivered packets is calculated as follows:

$$\text{percentage delivered packets} = \frac{\text{total number of delivered packets}}{\text{total number of sent packets}}$$

2. *Overhead*: This represents the overall number of packets sent. The overhead is calculated as the product of the total length of all paths in use, and the number of packets sent per second.

4.3.4 Experimental results

Several experiments were conducted to assess the performance of WEFTR (the proposed algorithm) with respect to the benchmark algorithms. The results of the following three sets of experiments are presented below:

- Variation in pause time
- Variation in sparsity
- Variation in faultiness of nodes

Variation in pause time As noted earlier, the pause time is a parameter specific to the simulation, which indicates how much an algorithm is capable of accommodating the mobility of the nodes. As seen from Fig. 1, we notice that just with respect to the overhead, the blind multipath routing is the worst, the DSR the best, and the E2FT somewhere in between the DSR and multipath curves. This is, of course, understandable. Our proposed algorithm further improves on the performance of E2FT by decreasing the overhead by 25–50%. For example, when the pause time is 250 seconds, the overhead for the multipath routing is 19,790, that for E2FT is 8,740, and for WEFTR it is 7,225. On the other hand, from Fig. 2, we observe that WEFTR achieves an almost similar order of performance when compared to E2FT. By examining Figs. 1 and 2 together, we can infer that our proposed algorithm (WEFTR) is capable of significantly reducing the overhead of the currently best available fault tolerant routing algorithm (E2FT), while achieving a performance packet delivery guarantee of at least 80%. Considering both these issues, it is clear that our algorithm always performs much better than both the DSR and the blind multipath routing schemes.

Variation in sparsity In the second set of experiments, we intended to study how the algorithms compare with respect to one another with the variation in the Sparsity of the nodes in the network. The value of Sparsity ranges between 0 and 1, where 0 represents the least percentage of

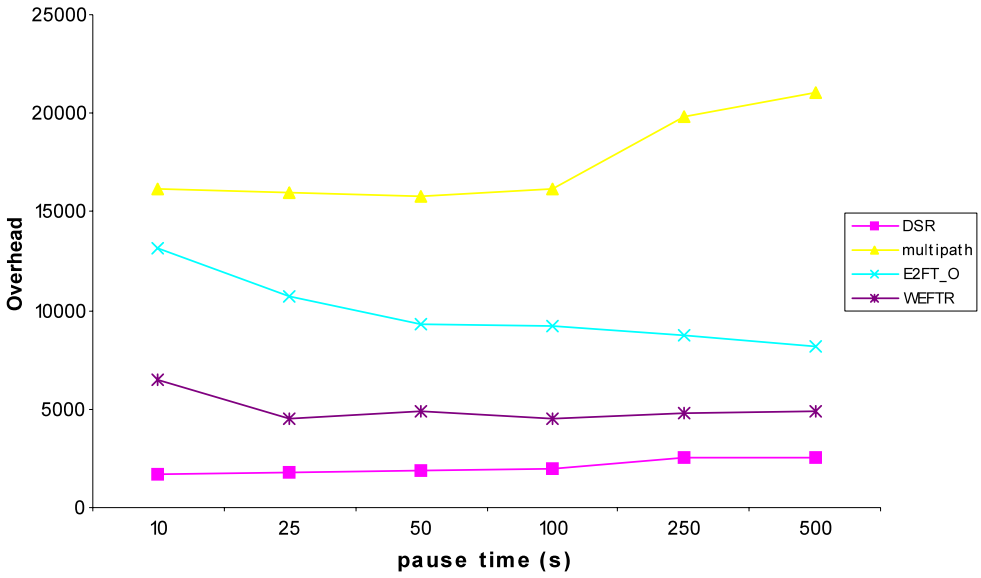

Fig. 1 Plot of overhead versus pause time for the various algorithms tested

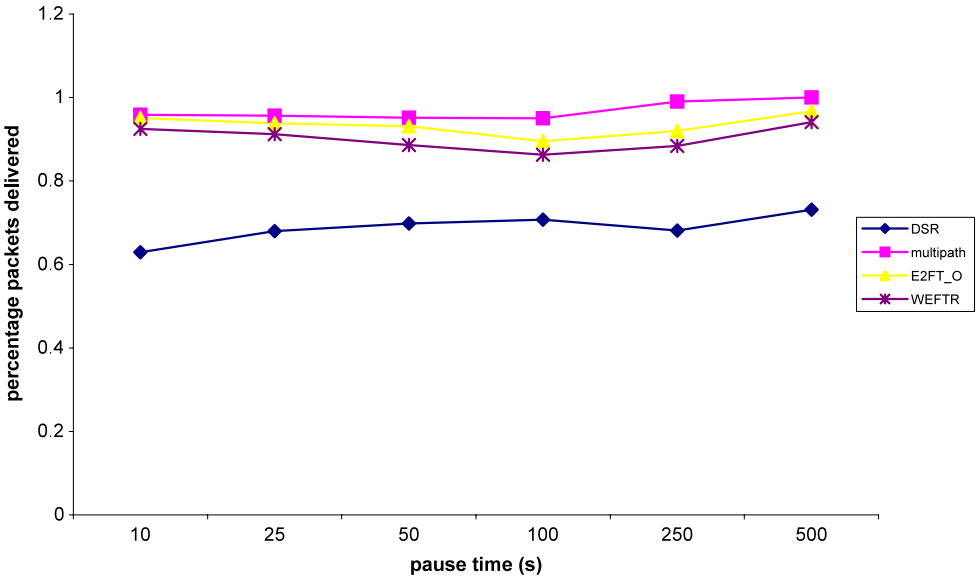

Fig. 2 Plot of percentage packets delivered versus pause time for the various algorithms tested

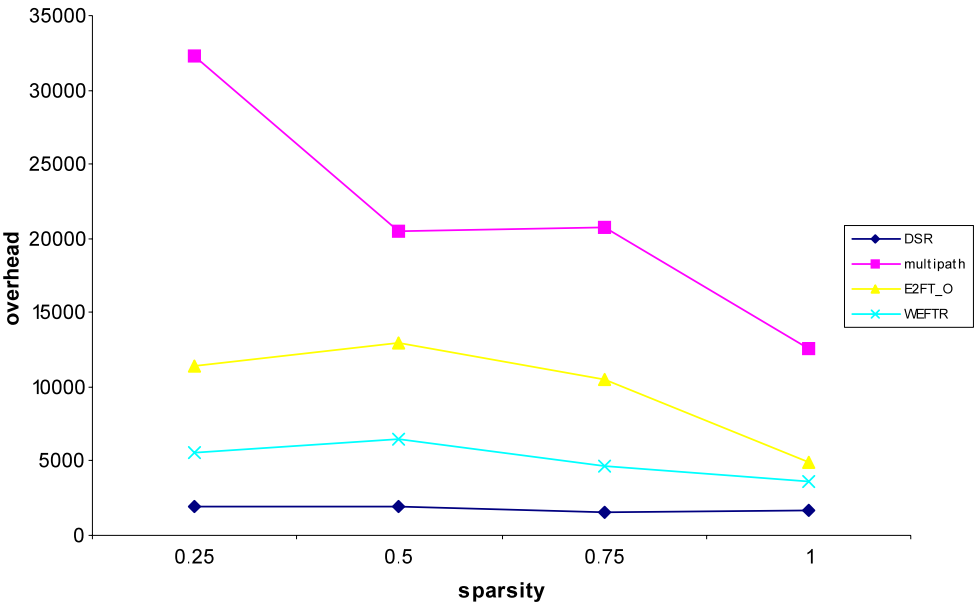

Fig. 3 Plot of overhead versus sparsity for the various algorithms tested

Sparsity, and 1 represents the greatest percentage of Sparsity. Since the nodes are mobile, how often they connect depends on how close they can get to one another, and this is thus directly related to the Sparsity. The different Sparsity values used in our experiments indicate the relative number of edges between the nodes in the network.

Figures 3 and 4 show the performance comparison of all the examined algorithms with respect to the overall overhead, and the percentage of packets successfully routed by the algorithms. From Fig. 3, we can clearly observe that even at different values of Sparsity, E2FT is capable of significantly reducing the overall overhead. For example, when the value of Sparsity is 0.25, the overhead for the multipath routing is 32,320, that of the E2FT scheme is 11,410. As opposed to this, the overhead for our proposed algorithm is 5,570. It should also be observed that the performance of E2FT is much better at lower Sparsity values than at the higher ones. On the other hand, if we look at Fig. 4, we can observe that, in general, the percentage of packets delivered by both E2FT and WEFTR are similar. Thus, for this set of experiments as well, we observe that WEFTR significantly reduces the overhead when compared to both the E2FT and blind multipath routing algorithms. This is done while achieving performance comparable to that of the E2FT or the schemes multipath (and certainly much better performance than that of the DSR algorithm) with respect to the number packets successfully routed.

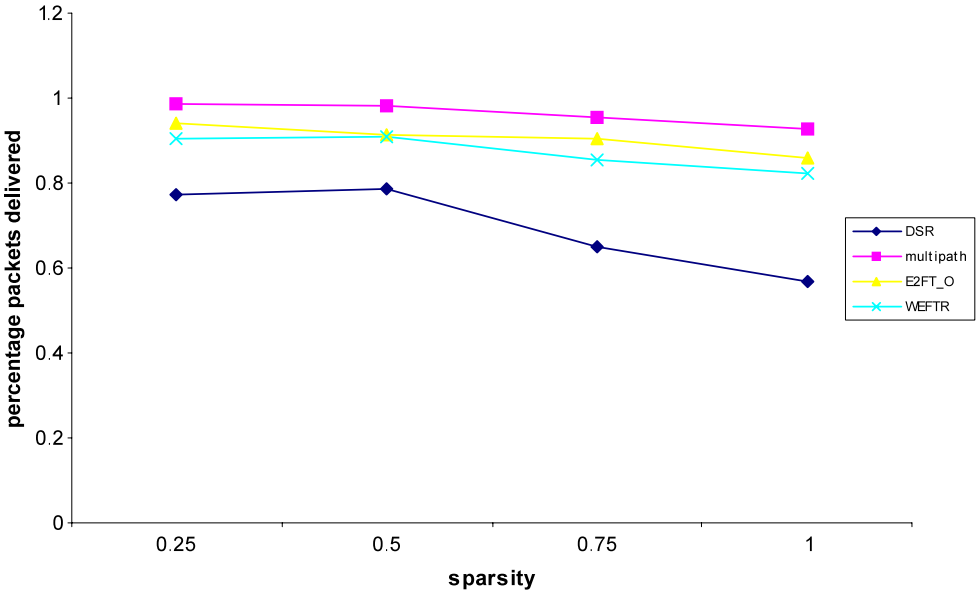

Fig. 4 Plot of percentage of packets delivered versus sparsity for the various algorithms tested

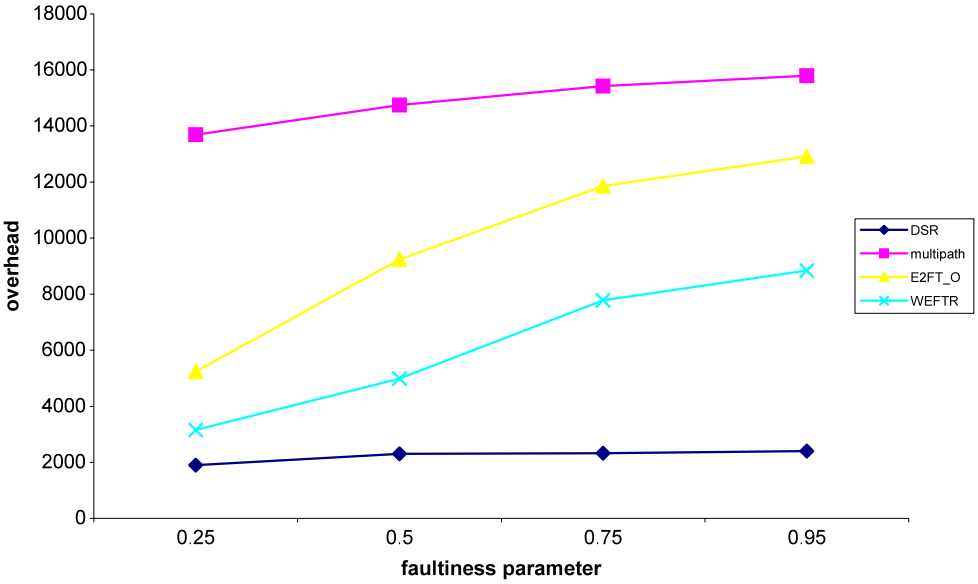

Fig. 5 Plot of overhead versus faultiness parameter for the various algorithms tested

Variation in faultiness Faultiness is an internal simulation parameter that indicates how many nodes will be faulty[8] in a given environment. It influences the faultiness behavior of the nodes, given their distance from the center of the region of operation of the nodes. Figures 5 and 6 show the variation in overhead, and the percentage of delivered packets, with the variation in the faultiness parameter. In our experiments, we have used the faultiness parameter to vary in a scale from a very low value to a very high value (i.e., on a scale of 0 to 1). We observe that, even in this set of experiments, our proposed algorithm demonstrates a much better performance as compared to the other algorithms. For example, at the faultiness parameter value of 0.25, the overhead for blind multipath routing is 13,690, for the E2FT it is 5,240, while it is 3,150 for the case of the WEFTR. Thus, in this case, our algorithm shows an improvement of about 62% over multipath routing, and an improvement of about 40% over E2FT algorithm. All of these algorithms, however, in general, show comparable performance with respect to the percentage of successfully delivered packets.

5 Conclusions

In this paper we have reported the results of the study of an interesting, yet challenging, problem of fault tolerant rout-

[8] In our simulations, we assume that the faulty nodes do not deliver any packets at all.

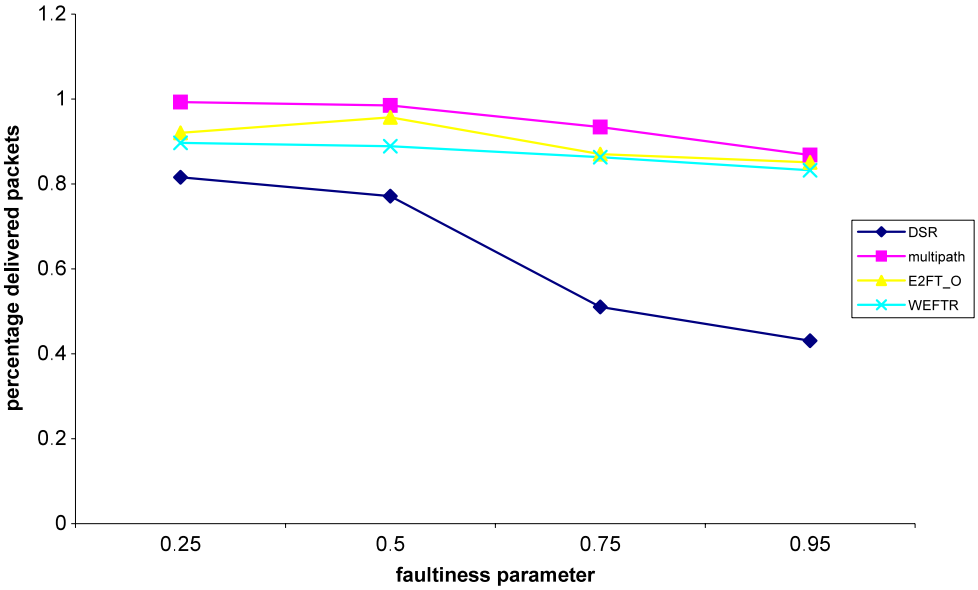

Fig. 6 Plot of percentage of delivered packets versus faultiness parameter for the various algorithms tested

ing in MANETs. The problem is that of *efficiently* routing packets in MANETs, in adversarial environments particularly, when there are misbehaving nodes present in the network. Thus, in essence, we have attempted to devise an algorithm, which is able to successfully route packets by "tolerating" faults in the network. There are two principal metrics that characterize any fault tolerant routing algorithm designed for MANETs: (1) The overhead, and (2) The percentage of successfully delivered packets. The traditional algorithms, the DSR and the multipath routing, have the potential to attain two extremes of each goal metric. While the multipath routing is a very strong algorithm for maximizing the number of successfully delivered packets, it introduces an extremely large overhead on the network. On the other hand, the DSR has a low overhead, but it simultaneously is a very poor fault tolerant routing algorithm, because it will drop packets if there are problems in the route identified by the algorithm. The E2FT algorithm proposed by Xue and Nahrstedt [10] is capable of minimizing the overhead when compared to the multipath routing algorithm, while achieving a similar order of performance (slightly inferior, to be precise) with respect to the number of packets successfully delivered.

Since the nodes are mobile, it turns out that the random variables encountered are non-stationary, implying that estimation methods for stationary variables are inadequate. Consequently, in this paper, we have presented a new fault-tolerant routing scheme that invokes a stochastic learning-based weak estimation procedure to enhance a route estimation phase, which, in turn, is then incorporated in a route selection phase. Our algorithm significantly reduces the overhead over the E2FT algorithm, while achieving a performance comparable (as far as the number of successfully delivered packets are concerned), and simultaneously minimizing the overhead. By rigorous simulation, we have shown that our algorithm was successful in achieving the above goal.

In the future, we intend to test our proposed scheme on more realistic networks and topologies, and to also consider how *alternate* sequence-based estimates can also be utilized advantageously to solve the same problem.

References

1. Bickel, P., & Doksum, K. (2000). *Mathematical statistics: basic ideas and selected topics* (Vol. 1, 2nd ed.). New York: Prentice Hall.
2. Caro, G. D., Ducatelle, F., & Gambardella, L. M. (2004). *AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks*. Technical Report No. IDSIA-25-04-2004, Dalle Molle Institute for Artificial Intelligence, Switzerland, August 2004. (Also appeared in *The Proceedings of parallel problem solving from nature VIII*. LNCS (Vol. 3242, pp. 461–470), Springer, Berlin, 2004.
3. Duda, R., Hart, P., & Stork, D. (2000). *Pattern classification* (2nd ed.). New York: Wiley.
4. Herbich, R. (2001). *Learning kernel classifiers: theory and algorithms*. Cambridge: MIT Press.
5. Johnson, D. B., & Maltz, D. A. (1996). Dynamic source routing in ad hoc wireless networks. In *Mobile computing* (pp. 153–181). Dordrecht: Kluwer Academic.
6. Marina, M. K., & Das, S. R. (2001). On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of the 9th international conference on network protocols*, Riverside, California (pp. 14–23).
7. Mueller, S., Tsang, R.P., & Ghosal, D. (2004). Multipath routing in mobile ad hoc networks: issues and challenges. In M. C. Calzarossa & E. Gelenbe (Eds.), *Lecture notes in computer science* (Vol. 2964). Berlin: Springer.
8. Nelakuditi, S., & Zhang, Z.-L. (2001). On selection of paths for multipath routing. In *LNCS: Vol. 2092. Proceedings of the 9th international workshop on quality of service*. London: Springer.
9. Perkins, C. E., & Royer, E. M. (1999). Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE workshop*

on mobile computing systems and applications, New Orleans, Louisiana (pp. 207–218).
10. Xue, Y., & Nahrstedt, K. (2003). Fault tolerant routing in mobile ad hoc networks. In *Proceedings of the IEEE wireless communications and networking conference* (*WCNC*), New Orleans, Louisiana, March 2003 (pp. 1174–1179).
11. Xue, Y., & Nahrstedt, K. (2004). Providing fault-tolerant ad hoc routing service in adversarial environments. *Wireless Personal Communications*, *29*, 367–388.
12. Oommen, B. J., & Rueda, L. (2006). Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments. *Pattern Recognition*, *39*, 328–341.
13. Oommen, B. J., & Rueda, L. (2004). A new family of weak estimators for training in non-stationary distributions. In *Proceedings of the 2004 international symposium on structural, syntactic, and statistical pattern recognition*, Lisbon, Portugal, August 2004 (pp. 644–652).
14. Wu, K., & Harms, J. (2001). On-demand multipath routing for mobile ad hoc networks. In *Proceedings of EMPCC*, Vienna, February 2001 (pp. 1–7).

B. John Oommen was born in Coonoor, India on September 9, 1953. He obtained his B.Tech. degree from the Indian Institute of Technology, Madras, India in 1975. He obtained his M.E. from the Indian Institute of Science in Bangalore, India in 1977. He then went on for his M.S. and Ph.D. which he obtained from Purdue University, in West Lafayettte, Indiana in 1979 and 1982 respectively. He joined the School of Computer Science at Carleton University in Ottawa, Canada, in the 1981–82 academic year. He is still at Carleton and holds the rank of a Full Professor. Since July 2006, he has been awarded the honorary rank of *Chancellor's Professor*, which is a lifetime award from Carleton University. His research interests include Automata Learning, Adaptive Data Structures, Statistical and Syntactic Pattern Recognition, Stochastic Algorithms and Partitioning Algorithms. He is the author of more than 320 refereed journal and conference publications, and is a *Fellow of the IEEE* and a *Fellow of the IAPR*. Dr. Oommen is on the Editorial Board of the *IEEE Transactions on Systems, Man and Cybernetics*, and *Pattern Recognition*.

Sudip Misra is a Doctoral student at Carleton University, in Ottawa, Canada. Prior to this, he received his Masters and Bachelors Degrees from the University of New Brunswick (Fredericton, Canada), and the Indian Institute of Technology (Kharagpur, India) respectively. He has several years of experience working in academia, government and the private sectors. Sudip has worked in R&D projects in project management, architecture, software design, and product engineering roles at Nortel Networks (Ottawa, Canada), Atreus Systems Corporation (Ottawa, Canada), and the Government of Ontario (Toronto, Canada). His current research interests include algorithm design and experimentation for high-performance, and high-speed Telecommunication Networks (with a special inclination towards technologies like SNMP, ATM, MPLS/GMPLS, and DWDM for optical networks). His other research interests include Learning Automata, Empirical Software Engineering, and Software Quality. He has published several research papers in journals and international conferences, and also received a few research awards.