Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

A Proactive Link-Failure Resilient Routing Protocol for MANETs based on Reinforcement Learning

Guido Oddi, Donato Macone, Antonio Pietrabissa and Francesco Liberati

Abstract — **Mobile-Ad-Hoc-Networks (MANET) are self-configuring networks of mobile nodes, which communicate through wireless links. One of the main issues in MANETs is the mobility of the network nodes: routing protocols should explicitly consider network changes into the algorithm design. MANETs are particularly suited to guarantee connectivity in disaster relief scenarios, which are often impaired by the absence of network infrastructures. This work proposes a proactive routing protocol, developed via Reinforcement Learning (RL) techniques, to dynamically choose the most stable path, basing on GPS information, among the feasible ones and to consequently increase resiliency to link failures. Simulations show the effectiveness of the proposed protocol, through comparison with the Optimized Link State Routing (OLSR) protocol.**

I. Introduction

Mobile-Ad-Hoc-Networks (MANET) are self-configuring networks of mobile nodes which communicate by wireless links. Since the network topology continuously varies due to the node mobility, the main challenge in MANET management is to allow each node to correctly route the packets to the other nodes. Therefore, it is crucial that MANET routing protocols consider network changes into the algorithm design itself, by explicitly taking into account information on mobility and link availability. MANETs are particularly suited in disaster scenarios, since they do not rely on fixed infrastructures. In this respect, the work presented in this paper was carried out in the EU funded project MONET [8], where a hybrid MANET-satellite network is being developed for disaster relief purposes. In rescue scenarios, as the ones encompassed in MONET, the communications should be resilient to link failures due to the mobility of the nodes, which cause the impossibility to exchange vital information between rescue teams. The routing protocol proposed in this paper directly considers such issue into the algorithm design, taking advantage from GPS information.

Routing in MANETs is a widely studied research topic, and the problem has been dealt with by a plethora of approaches (see for instance [11] and [12] and the references therein). Three main classes of routing algorithms are studied: proactive, reactive and hybrid algorithms. Proactive algorithms aim at setting up in advance all source-destination paths within the network, regardless the necessity of forwarding a data traffic flow to a specific destination node. In reactive algorithms, source-destination paths are calculated on-demand, i.e., as soon as a data traffic flow needs to be transmitted. Hybrid algorithms jointly use both the approaches. Two protocols are widely used in literature in this field and are usually considered as benchmarks for subsequent algorithms: the Optimized Link-State Routing Protocol (OLSR) [5] and the Ad hoc On-Demand Distance Vector protocol (AODV) [6]. The OLSR protocol is a proactive protocol based on the link state approach [11]: each node internally generates a network connectivity graph and associates a specific information to each link of the graph representing its current state; each node independently calculates the best route towards each feasible destination. OLSR uses an optimized flooding to disseminate control and topology information to all network nodes (Hello and Topology Control – TC messages) through the concept of Multipoint Relays (MPR). See [11] for further details. AODV is a reactive routing protocol based on the distance vector approach [12]: each node is not aware of the whole network topology, but it associates to each destination one value representing the distance to reach the destination, expressed by a proper metric (e.g., number of hops), and the indication of the related next-hop. AODV avoids the loop formation, which occurs in MANETs due to variability of the network graph [11]. Signalling includes the broadcasting of specific Route Request (RREQ) and Route Reply (RREP) messages, which generate the requested source-destination path, and Route Error (RERR) messages for link failure management purposes. Similarly to the OLSR, the algorithm proposed in this paper follows a proactive approach, constantly updating network control information in a distributed fashion. However, the proposed algorithm also presents similarities with the AODV protocol, since the approach is similar to the distance vector one.

The proposed routing algorithm is based on the Q-Routing algorithm [2] developed for fixed networks, and extends it to cope with mobile scenarios, considering proper link stability metrics; therefore, the algorithm is named Link Failure Resilient Q-Routing (FQ-Routing). In particular, FQ-Routing is based on Reinforcement Learning (RL) methodologies. GPS information are assumed to be exchanged among MANET devices, in compliance with the specifications of the mobile nodes developed within MONET project. The main concept of the proposed algorithm is that the routing strategy consists of proactively

Manuscript received January 30, 2012. This work was supported by the EU funded project MONET under Grant Agreement no. 247176.

G. Oddi is with the University of Rome "Sapienza", Rome, 00185, Italy (corresponding author – phone: +390677274037; fax: +390677274039; e-mail: oddi@dis.uniroma1.it).

D. Macone, A. Pietrabissa and F. Liberati are with the University of Rome "Sapienza", Rome, 00184, Italy (e-mail addresses: macone@infocom.uniroma1.it, {pietrabissa, liberati}@dis.uniroma1.it).

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

choosing the next-hop node in order to use the most stable path, taking into account link failure prediction, computed on the basis on GPS information. This paper is organized as follows: Section II details RL and Q-Routing; Section III reports some modifications to cope with critical MANET scenarios, taking into account mobility and link stability. Section IV presents the simulation results, and in Section V the conclusions are drawn.

II. BACKGROUND

A. Reinforcement Learning (RL)

Under the Markovian[1] and stationarity assumptions, a discrete-time Markov Decision Process (MDP) is defined by: i) a finite state space $S = \{s_i\}_{i=1,\dots,n}$; ii) a finite and non-empty set of available control actions $A(s_i) = \{a_k\}_{k=1,\dots,|A(s_i)|}$ associated to each state $s_i \in S$; iii) a real-valued one-step reward function $r: S \times A \to \mathbb{R}$, where $r = r(s_i,a_k)$ is the reward incurred by the system as it is in state $s_i \in S$ and action $a_k \in A(s_i)$ is chosen; iv) the probability $p(s_i,s_j,a_k)$ that, in the next time step, the system will be in state $s_j \in S$ when action $a_k \in A(s_i)$ in state $s_i \in S$ is chosen; the set of these transition probabilities constitute the transition matrix **T**. In the following, we will denote with $s(t)$, $a(t)$, and $r(t)$ the state, the action and the reward of the system at time t, respectively. A stationary policy is a function $\pi: S \to A$, which maps every state $s_i \in S$ to a unique control action $a_k \in A(s_i)$. When the system operates under policy π, the MDP reduces to a discrete-time Markov chain, and the following expected discounted reward is earned:

$$R_\pi = \limsup_{n\to\infty} \frac{1}{t} E_\pi \left\{ \sum_{t'=1}^{t} \gamma^{t'-1} r(t') \right\}, \quad (1)$$

where $0 \le \gamma \le 1$ is the discount rate and the subscript π specifies that the controller operates under policy π. The MDP problem is the determination of the optimal policy π^* minimizing cost (1). Beside standard Dynamic and Linear Programming approaches [9], MDPs can be solved also via RL algorithms [1]: even if RL approaches achieve the optimal solution only under given conditions, such as infinite number of visits of each state, in practice they achieve approximate solutions to problems which are intractable for other methods. In our scenario, the RL approach was chosen since the Dynamic and Linear Programming approaches are not suited, mainly because (i) they need complete knowledge of the system statistical characteristics (i.e., the elements of transition matrix **T**); (ii) they cannot cope with non-stationary scenarios. RL algorithms are based on the observations of transition occurrences and gained rewards: in a RL task, in which the agent and the environment interact at discrete time steps, at time t, the agent observes state $s(t)$ and produces an action $a(t)$; then, it receives the reward $r(t+1)$ and observes the environment to infer the new state $s(t+1)$. The agent's objective is to learn an optimal policy π^*, such that the discounted return (1) is maximized.

The Q-learning algorithm, a simple and effective RL algorithm, is now briefly described. Under a given policy π, each pair (s_i,a_k), $s_i \in S$, $a_k \in A(s_i)$, has an associated action-value function $Q_\pi(s_i,a_k)$, which represents the reward to execute action a_k in state s_i, summed to the expected reward achieved by following policy π, starting from the next state. The Q-learning learns the action-value functions on-line, with the following update rule:

$$Q(s(t),a(t)) \leftarrow Q(s(t),a(t)) + \alpha[r(t+1) + \gamma \cdot \max_{a_k \in A(s(t+1))} Q(s(t+1),a_k) - Q(s(t),a(t))], \quad (2)$$

where α is the learning rate of the agent, which has to be tuned based on the specific application scenario and allows to account for non-stationary scenarios. The convergence to the optimal policy is guaranteed if $0 \le \gamma < 1$ and all actions and state are "sufficiently" visited ([1]). Given the estimate (2) of the action-value function $Q(s_i,a_k)$, an associated policy can be computed in different ways. The 'greedy' policy is determined by selecting in each state $s_i \in S$ the best (*greedy*) action, i.e. the action a_k associated with the larger value according to the action-value function. However, to favour the learning process of RL, sometimes different actions should be selected in order to explore the state space; for this purpose, some algorithms use the so-called *ε-greedy* policy, determined by selecting in each state $s_i \in S$ the greedy action with probability 1-ε, and a random action with probability ε, with $0 < \varepsilon \le 1$,:

$$a^{\varepsilon-greedy}(s_i) = \begin{cases} \arg\max_{a_k \in A(s_i)} Q(s_i,a_k), \text{ with } \Pr = 1-\varepsilon \\ \text{random action in } A(s_i), \text{ with } \Pr = \varepsilon \end{cases}, \forall s_i \in S, \quad (3)$$

The tuning of the parameter ε is needed to balance exploration (i.e., random action selection) and exploitation (i.e., greedy action selection based on the current best known policy). The Q-learning approach consists in learning the action-value function Q, and therefore the optimal policy, by experience, following proper update rules, starting from a given initial Q function (e.g., with all values set to 0).

B. Q-Routing

As already mentioned, the proposed routing technique is an extension of Q-Routing ([2], [3]). In the Q-Routing algorithm, the network is modelled as a graph $G = (V,E)$, where V is the set of nodes with cardinality $|V| = n$ and E is the set of edges. Each node $i \in V$ is connected to a set of neighbour nodes N_i. The network routing problem is formulated as a MDP, considering a single packet coming from a source s and directed to a destination d: $S = \{s_i = \delta_i \mid i \in V\}$, where δ_i is a vector of n elements equal to 0 but the i^{th} element equal to 1, is the finite state set; each state indicates the (unique) current position of the packet;

[1] A stochastic process has the Markov property if the conditional probability distribution of the next state of the process depends only upon the present state and is conditionally independent of past states.

$A = \bigcup_{i=1,\ldots,N} A(s_i)$ is the action set, where $A(s_i) = \{a_{ij} \mid j \in N_i)\}$ is the action set of state s_i, and a_{ij} represents the decision of the agent to move from state s_i to state s_j (i.e., node i sends the packet to neighbour j); the one-step reward r_j obtained performing action $a(t) = a_{ij}$ in state $s(t) = s_i$ to reach state $s(t+1) = s_j$ is 1 only if at time $t+1$ the packet is in the destination node d, it is 0 otherwise.

The transition matrix is not defined, since RL algorithms implicitly estimate the transition probabilities within the action-value function estimation (2). The routing algorithm runs as follows. Each node $i \in V$ keeps in memory a matrix of elements $Q_i(d, j) \in [0,1]^{n \times |A(s_i)|}$. $Q = \bigcup_{i \in N} Q_i$ is the action-value function: in node i, the routing decision (i.e., the next-hop $j \in N_i$ to reach destination $d \in V$) is taken following an ε-greedy policy, i.e., by choosing the neighbour such that $j = \arg\max_{l \in N(i)} Q_i(d,l)$ with probability (1 - ε) (exploitation), or randomly with probability ε (exploration); then, the system moves to state s_k. The update procedure (2) occurs every time a packet is delivered to a neighbour node, and requires the neighbour which receives the packet from node i to send back an acknowledge packet (ACK). Note that this definition of action-value function implicitly allows a distributed implementation of the algorithm, since i) each node i is responsible of updating its own part Q_i of the value function, and ii) the update rule (2) requires data from the neighbour nodes N_i only. If the network topology is stationary, as the update (2) of the Q function keeps going on, an estimation of the real state of the network is achieved: thanks to the reward r_j definition, each element $Q_i(d,l)$ represents the discounted reward incurred by the packet to reach node $d \in V$ when it is in node i following a path through neighbour $j \in N_i$. The Q-Routing algorithm has the desired property of converging to the shortest path based on the hop-count metric: due to the fact that $\gamma < 1$, the longest path has the smallest action-value; furthermore, the random exploration entailed by the ε-greedy action selection allows to effectively update the action-values.

III. FQ-ROUTING ALGORITHM

By analyzing the Q-Routing behavior in mobile applications and, in particular, in critical scenarios as the ones considered in the MONET project, the following main drawbacks were identified: i) no adaptability to variable scenarios and slow reactions to path errors due to link failures: the ε-greedy random exploration is not sufficient to follow rapid network changes due to node mobility and changes of node/network resources; ii) ACK messages generates an unacceptable amount of overhead; iii) loop management (i.e., the detection and resolution of loops in paths followed by packets), crucial in mobile networks, is not considered. The modifications of the Q-Routing algorithm aim at solving the listed drawbacks by reporting a set of appropriate extensions. In particular, Section III.A addresses the first point by defining a variable discount factor, Section III.B addresses the second and third points by introducing a proactive approach, and Section III.C the fourth point by introducing a detection-based loop management. Section III.B describes the algorithm properties.

A. Link stability metrics

To take into account the mobility during the routing process, proper metrics are described in this Section. The idea follows the one introduced in Wu et al. [3], which defines a time-varying discount factor $\gamma = \gamma(t)$. In FQ-Routing, to account for non-homogeneous characteristics of the network links and nodes, different time-varying discount factors are associated to different nodes. The update rule (2) is modified accordingly:

$$Q_i(d, j) \leftarrow Q_i(d, j) + \alpha[r_j + \gamma_{ij}(t) \cdot \max_{l \in N(j)} Q_j(d,l) - Q_i(d, j)] \quad (4)$$

where $\gamma_{ij}(t)$ is the value of the discount factor at time t, when the system is in state $s(t) = s_i$ and moves to state $s(t+1) = s_j$ under action $a(t) = a_{ij}$. The initial values of the state-value function are assumed to be equal to 0. Two different metrics are considered, taking into account the link availability prediction and the node mobility (this second metric follows the idea proposed in [3]). A link availability metric is proposed. It is computed by assuming that the nodes are GPS-enabled, and is used to compute the link availability factor $\gamma_{ij}^{GPS}(t)$. Such metric uses the concept of 'neighbour reachability time', as defined in [7]. This time is calculated as follows. Let $[x_i(t),y_i(t)]$ and $[x_j(t),y_j(t)]$ be the positions at time t of nodes i and j, respectively, on a previously defined Cartesian coordinate system. Let also $v_i(t)$ and $v_j(t)$ be the velocities of the nodes at time t, $\theta_i(t)$ and $\theta_j(t)$ be their moving directions and TX be their transmission range. Then, the amount of time $T_{ij}(t)$ the mobile nodes i and j will stay connected, given the GPS information received at instant t, is given by:

$$T_{ij}(t) = \frac{-(ab+cd) + \sqrt{(a^2+c^2)TX^2 - (ad-cb)^2}}{(a^2+c^2)}, \quad (5)$$

where $a = v_i(t)\cos[\theta_i(t)] - v_j(t)\cos[\theta_j(t)]$, $b = x_i(t) - x_j(t)$, $c = v_i(t)\sin[\theta_i(t)] - v_j(t)\sin[\theta_j(t)]$ and $d = y_i(t) - y_j(t)$. Once such a neighbour-reachability time is calculated, the time-varying discount factor is generated in the following way:

$$\gamma_{ij}^{GPS}(t) = \begin{cases} \sqrt{\dfrac{T_{ij}(t)}{T_{HELLO}}} & T_{ij}(t) < T_{HELLO} \\ 1 & T_{ij}(t) \geq T_{HELLO} \end{cases}. \quad (6)$$

In practice, equation (6) is representative of the probability that the link between nodes i and j is available T_{HELLO} seconds after the reception of a *Hello Packet* (see Section III.B). In such a way, links which are supposed to be stable in the next period will be chosen with higher probabilities. To let node i compute the link availability

factor (6), each neighbour node $j \in N_i$ must communicate its position, speed and direction to node i. The main goal of metric (6) is to avoid the loss of a consistent portion of traffic in the time necessary to manage a link failure due to node mobility by changing the path *before* the link failure occurrence. The assumption under equation (5) is that nodes have simple mobility patterns (for example, quite-constant speed and no sudden changes of direction with brusque accelerations). In the considered scenario, the movement of a team of rescuers is the most of time in groups towards sensible objectives at quite constant speed with small accelerations, so that the model can be applied with sufficient accuracy.

To enhance the link stability estimation, and to broaden the addressed situations and scenarios, a mobility metric $\gamma_j^{MOB}(t)$ is considered, which accounts for the degree of node mobility. The mobility metric follows the idea proposed in [3], which takes into consideration the mobility of nodes in terms of neighbour sets. Let $A \Delta B$ denote the symmetric difference between two sets A and B, and $A \cup B$ denote the union of these sets. The mobility factor is then calculated as the percentage of neighbours which remains the same between the sending of two consecutive hello packets:

$$\gamma_j^{MOB}(t) = \sqrt{1 - \frac{|N_j(t) \Delta N_j(t - T_{HELLO})|}{|N_j(t) \cup N_j(t - T_{HELLO})|}}, \qquad (7)$$

where, with little abuse of notation, $N_j(t)$ is the set of neighbours of node j at time t and T_{HELLO} is the hello interval. The mobility factor (7) is computed by each node and sent to its neighbours; node i is then capable of computing $\gamma_j^{MOB}(t)$ since it receives the value of the mobility factors of each candidate next-hop nodes $j \in N_i$. The main importance of the mobility metric is to let the routing algorithm prefer more stable nodes, with a high value of mobility factor, which are not likely to rapidly change their neighbour sets.

Note that (i) the mobility factor is a per-node factor, whereas the link availability factor is a per-link factor (involving the couple of nodes originating the link), and that (ii) together, the link availability and the mobility metrics provide information on the link stability (i.e., the stability of the communication between two nodes), based on their relative position and speed (link availability metric) and on node j behaviour (mobility metric). The two metrics are then combined together to yield the overall time-varying discount factor, from state s_i and state s_j, computed as: $\gamma_{ij}(t) = \gamma^{MAX} \cdot \gamma_{ij}^{GPS}(t) \cdot \gamma_j^{MOB}(t)$. The square root in equation (6) and (7) is used to avoid to excessively lower the whole discount factor. The two factors $\gamma_{ij}^{GPS}(t)$ and $\gamma_j^{MOB}(t)$ are, by definition, normalized between 0 and 1 and $\gamma^{MAX} \in [0,1)$ represents the maximum discount factor (this value is included in $\gamma_{ij}(t)$ since, in case the two variable factors $\gamma_{ij}^{GPS}(t)$ and $\gamma_j^{MOB}(t)$ are equal to 1, i.e., with fixed nodes, the algorithm is guaranteed to tend towards convergence, as mentioned in Section II); clearly, since the overall discount factor is variable, the convergence paths change in time. The whole discount factor $\gamma_{ij}(t)$ is used in equation (4) to update $Q_i(d,j)$, i.e., the part of the action-value function which is handled by node i. Each single metric gives a contribution to the total metric in relation to its current importance. Since FQ-Routing is a distributed algorithm (property inherited from the Q-Routing algorithm), each neighbour node $j \in N_i$ must communicate some information to node i to let it properly compute the variable discount factor, as described in the following section.

B. Proactive approach

To adapt to network changes and to decrease the overhead, the update rule (2) and the exploration are performed in a *proactive* fashion (e.g. as in OLSR), upon the receipt of a *Hello Packet* (ACKs are no longer needed). The interval T_{HELLO} between two hello messages is the duration of the time step of the routing algorithm and addresses the trade-off between overhead and reactivity. At each time step, each node broadcasts to its neighbours a *Hello Packet* containing (i) its maximum estimates of the action-values towards each destination node, and (ii) all the information needed to calculate the variable discount factor, as detailed in Section III.A. This periodic message exchange gives the algorithm the following advantages with respect to the Q-Routing: (i) reduction of the number of protocol control messages, since no per-packet ACK messages are needed and no random exploration is foreseen (as in ε-greedy policy instead); (ii) adaptation to variable scenarios, since the proactive approach maintains the action-values as much up to date as possible by periodic exploration, whereas the ε-greedy policy relies on a random exploration, which is unable to rapidly follow network changes without causing an unacceptable throughput decrease. Moreover, the proactive approach has been chosen to decrease the path setup delay of each flow, especially important in rescue scenarios, in which critical (voice and video/data) communications among users must be setup immediately at the request.

C. Loop management

Loops can be generated because of a link failure between two nodes: the obsolescence of routing information inside different nodes can lead to the formation of a loop (see [6], for further details). In common distance vector protocols for MANETs, such as AODV [6], the path towards a destination is unique and univocal and a sequence number, representing the obsolescence of a path, can be directly associated to it. In FQ-Routing, in contrast, several overlapping paths to a destination can coexist, since the best paths vary over time (according to the node mobility and to the time-varying discount factors), leading to the difficulty to manage loops with mono-dimensional sequence numbers. Thus, in FQ-Routing, loops are managed in a *detection-based* way: 1) as soon as a loop is detected in node n, through the inspection of the sequence number included in a *Data Packet*, it sends a

Probe Packet towards destination *d*, following the path determined by the current routing tables; 2) if the *Probe Packet* reaches again node *n*, then a *Path Reset* message is forwarded following the reverse loop path determined by the *Probe Packet* journey (this reverse path represents a loop); 3) when a loop node receives the *Path Reset* message, it resets to zero the associated action-values and broadcasts new next-hop *n'*, if available, and the associated $Q_s(d,n')$, to speed up the update of routing tables.

D. FQ-Routing properties

At first sight, the FQ-Routing approach seems to be very similar to the common distance vector approaches, but it presents some key differences. In each node, distance vector protocols associate to each destination a single value, representing the distance to reach a determined destination via the selected next-hop, and only one path a time is available and exploited. The consequence is that the algorithms are not effective in switching to a different next-hop (handover), and also in balancing the traffic with respect to a specific metric. Therefore, distance vector approaches are not properly adequate for mobile scenarios. The FQ-Routing strategy, on the contrary, maintains for each destination a vector of values, each one associated to a next-hop node. This helps in performing fast handovers and allows exploiting different paths as network conditions vary. In fact, the best paths vary over time according to the node mobility and to the time-varying discount factors. In conclusion, the FQ-Routing protocol has *fault-tolerant* characteristics: if a link failure occurs, a new path is immediately set up, since every node keeps estimates of the conditions of all the available paths; moreover, every node tries to predict when a link failure occurs (via the link stability metrics, which account for the node mobility) in order to trigger path changes even before a link fault occurs.

IV. Simulations

The FQ-Routing protocol was implemented in C and simulated using OPNET 16.0 [4] environment.

TABLE I
Mobility Model

Attribute	Value
Type	Random Waypoint
x_min [m]	0
y_min [m]	0
x_max [m]	Scenario dependent
y_max [m]	Scenario dependent
Speed [m/s]	Scenario dependent
Pause Time [s]	constant(5)
Start Time [s]	constant(30)
Stop Time [s]	End of simulation

Proof of concepts simulations, taking into consideration the link stability metrics, are reported where the FQ-Routing is compared to standard OLSR [5] protocol. All simulation parameters are in line with the actual civil protection requirements collected within the MONET project [14]. Table I and Table II define the mobility model (namely, the Random Waypoint mobility model [10]) and the routing protocols parameters (both for OLSR and for FQ-Routing) used in the simulations.

TABLE II
Routing Protocols Parameters

Attribute	Value
OLSR Willingness	Default (3)
OLSR Hello Interval [s] – TC Interval [s]	2 – 5
OLSR Neighbour Hold Time [s]	6
OLSR Topology Hold Time [s]	15
OLSR Duplicate Message Hold Time [s]	30
FQ-Routing γ^{MAX}	0.95
FQ-Routing α	0.95
FQ-Routing T_{HELLO} [s]	See sim.
FQ-Routing *Neighbour hold timeout* interval [s]	See sim.

The wireless link model used in all the simulations is the Wi-Fi 802.11a [13] at 54Mbps. The transmission range *TX* is 115m. Two simulation sets were carried out: i) GPS-aware link availability metric simulation, ii) throughput increase using link stability metrics. In these simulations, the T_{HELLO} and *Neighbour hold timeout interval* parameters of FQ-Routing are set equal to the *Hello Interval* and the *Neighbour hold timeout* parameters of OLSR.

A. GPS-aware link availability metric simulation

This section reports a network scenario, shown in Fig. 1, in which two alternative paths are present; due to mobility, one of the feasible paths becomes unavailable. A bidirectional Pulse Code Modulation (PCM) constant audio flow at 64kbps is set up between the Source and Destination nodes. The factor $\gamma_j^{MOB}(t)$ is identically set equal to 1, so that the overall discount factor only depends on the GPS-aware link

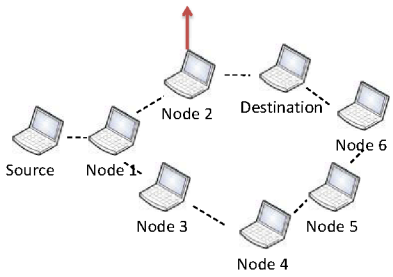

Fig. 1. GPS-aware example scenario; the arrow indicates the direction of Node 2, which is leaving the network

availability metric: $\gamma_{ij}(t) = \gamma^{MAX} \cdot \gamma_{ij}^{GPS}(t)$. There exist two different paths between Source and Destination nodes, the shortest one via nodes 1 and 2 and the second one via nodes 1-3-4-5-6. Node 2 starts to move from its initial position at time 120s and reaches the transmission range limit of 115m after 190s, moving with a constant speed of 2m/s. Fig. 2 shows the traffic received (at application level) at destination node, comparing OLSR with the FQ-Routing. OLSR needs to wait for Node 2 to reach the transmission limit *TX* to switch the path, and this happens only after a prefixed *Neighbour Hold Time* timeout expires, starting from the last *Hello Packet* received by Node 1. The GPS-aware FQ-Routing, in contrast, starts switching the path in advance, estimating that the link will soon go down: in this way, the

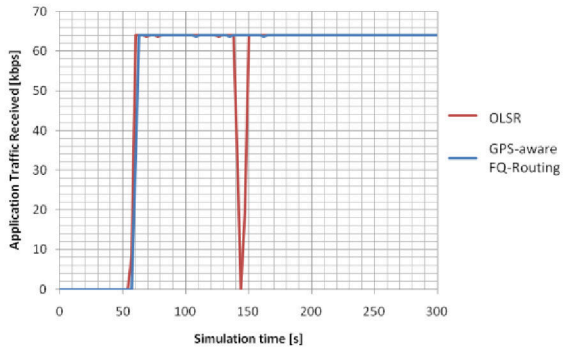

Fig. 2. Traffic Received comparison using GPS-aware metric

throughput of the received traffic does not suffer from the switching and from the consequent loss of packets (the packet loss in OLSR is about 73.6% in the 6 seconds needed for the switching procedure). Note that the Q-Routing suffers from the same drawbacks of OLSR, since both depend uniquely on the hello intervals and do not estimate link availability time. In this respect, the enhancement provide by the FQ-Routing is due to the new metric and to the introduction of AODV-like path error messages.

B. Throughput increase using link stability metrics

This Section reports the behaviour of the FQ-Routing when the node speed increases. The overall discount factor (described in Section III.A) depends on both the link stability metrics: $\gamma_{ij}(t) = \gamma_{ij}^{GPS}(t) \cdot \gamma_{j}^{MOB}(t)$. The simulated scenario is a square area of 400m x 400m, in which 20 nodes are deployed and move following a Random Waypoint mobility model [10] with a prefixed speed. The learning rate parameter α of the FQ-Routing approach is set to 0.99, very close to 1, in order to rapidly adapt to network changes. Four bidirectional GSM G.711 audio flows at 13.6kbps are set up between 4 network nodes. The simulations have a duration of 300s and were performed by varying the speed in the interval 1-15m/s (as proposed by MONET project in [14]). For each speed, the simulation was repeated 50 times. The application level normalized throughput (expressed as the percentage of traffic received at the application layer in destination node) was analyzed and the behaviour is reported in Fig. 3, where is also compared to OLSR. The figure shows that the link stability metrics consistently improve the algorithm behaviour at high speed. The throughput increase of the FQ-Routing compared to OLSR averaged along 1-15m/s speeds is 10.2% and reaches a peak of 25.9% at 15m/s. This result is intuitive because the link availability and the mobility metrics are more effective at high speeds, as the estimated node positions rapidly vary and the mobility factor assumes more importance.

V. CONCLUSION

This paper proposes a proactive version of the Q-Routing [2] algorithm, based on Reinforcement Learning, suitable for Mobile Ad-hoc Networks (MANETs), taking advantage from GPS information. In this respect, link stability metrics are considered to take into account link availability and mobility. The algorithm was compared with the Optimized

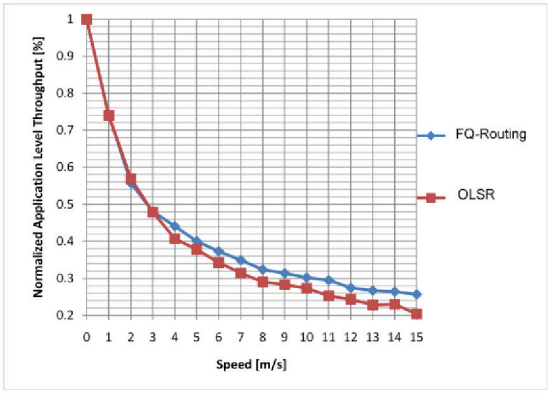

Fig. 3. Throughput comparison using link stability metrics

Link State Routing (OLSR) protocol and simulations show that the application level throughput and the link failure resiliency are increased. The proposed approach suggests a number of interesting directions for further work. For instance, the algorithm formulation allows seamless introduction of further metrics. A first example could be the introduction of one or more energy-aware metrics (focused on the residual battery inside the nodes) with the aim of balancing the battery consumption of nodes belonging to the whole MANET, useful especially in disaster relief scenarios.

REFERENCES

[1] R. S. Sutton and A. G. Barto, "*Reinforcement Learning – An Introduction*", 1998.

[2] J. A. Boyan, M. L. Littman, "*Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach*", in Advances in Neural Information Processing Systems, Vol. 6 (1994), pp. 671-678.

[3] C. Wu, K. Kumekawa and T. Kato, "*A MANET protocol considering link stability and bandwidth efficiency*", ICUMT2009, 12th-14th October 2009, St. Petersburg, Russia.

[4] *"OPNET Modeler 16.0"*, http://www.opnet.com/, consulted on May 2011.

[5] "*Optimized Link State Routing Protocol (OLSR)*", RFC3626, http://tools.ietf.org/html/rfc3626 consulted on May 2011.

[6] "*Ad hoc On-Demand Distance Vector (AODV) Routing*", http://www.ietf.org/rfc/rfc3561.txt, consulted on May 2011.

[7] W. Su, S-J. Lee and M. Gerla, "*Mobility prediction and routing in ad hoc wireless networks*", International Journal of Network Management, Vol. 11, 3-30.

[8] MONET (Mechanisms for Optimization of hybrid ad-hoc networks and satellite NETworks), FP7-ICT project (Contract no.: 247176), www.ict-monet.eu.

[9] M. L. Puterman, "*Markov Decision Processes*", New Jersey, Jhon Wiley & Sons, 1994.

[10] T. Camp, J. Boleng, V. Davies, "*A Survey of Mobility Models for Ad Hoc Network Research*", Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002

[11] S. Corson, J. Macker, "*Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*", IETF RFC2501, January 1999.

[12] L. Chen; W.B. Heinzelman, "*A Survey of Routing Protocols that Support QoS in Mobile Ad Hoc Networks*," Network, IEEE , vol.21, no.6, pp.30-38, November-December 2007.

[13] "*IEEE 802.11™ Wireless Local Area Networks: the Working Group for WLAN Standards*", available at http://www.ieee802.org/11/, accessed on October 2011.

[14] "*Deliverable 2.3 – MONET Requirements*", MONET FP7-ICT project, www.ict-monet.eu.