Evaluating a bound for MANETs routing protocols performance using graphs with activation windows

David Soler, Jose Albiach and Eulalia Martinez
Polytechnic University of Valencia
Dept. Applied Mathematics
Camino de Vera, s/n, 46071 Valencia, SPAIN
Email: dsoler@mat.upv.es

Pietro Manzoni
Polytechnic University of Valencia
Dept. Computer Engineering
Camino de Vera, s/n, 46071 Valencia, SPAIN
Email: pmanzoni@disca.upv.es

***Abstract*—In this paper we present an algorithm called STPA (Shortest Time Path Algorithm) which aims at providing a comparison tool for the evaluation of a bound for Mobile Ad Hoc Networks (MANETs) routing protocols performance.**

STPA provides an exhaustive evaluation of an ideal routing protocol. Based on the current position and state of the nodes it can determine factors like: how many complete messages get to the destination, which is the smallest amount of time required by a packet to get to the destination, which path followed each packet, and so on. This values would allow a protocol designer to improve or fine tune his proposal.

We demonstrate that the complexity of the algorithm is $O(\sigma^2)$, that is polynomial with respect to parameter σ; where σ corresponds to the sum of all the instants of time during which all nodes are active.

I. Introduction

Mobile ad-hoc networks (MANETs) are networks where mobile nodes can roam around at will and communicate with each other without any preexisting communication infrastructure.

A routing path has to be established in a multi-hop manner, requiring each mobile host to serve as a router. This fact creates many challenging research issues since collaborating nodes might have different capacities, like processing power, memory availability or willingness to be part of a route. This last factor, the *willingness to be part of a route* might depend on aspects such as the need for a node to be left physically turned off to save energy [1], or to simply make this node invisible, i.e., logically turned off, to certain other nodes for security reasons [2]. The effect of the presence of nodes in the off-state must therefore be thoroughly evaluated.

The evaluation of a new proposal is usually performed with the aid of a network simulator, like for example the ns-2 [3]. Anyway, the need for a formal model to describe a MANET has been approached in many other different ways. In [4] the author use games theory to evaluate cooperation in ad hoc networks for energy optimization. In [5] a framework called MERIT is described that can be used to assess routing protocols in MANETs. It uses the novel concept of a shortest mobile path (SMP) in a mobile graph, a generalization of the shortest path problem for mobile environments. Finally, in [6] the authors propose a novel graph-based mobility model, which provides a more realistic movement than the random walk model by reflecting the spatial constraints in the real world.

In this work we propose an analytical model based on graphs with activation windows. Our proposal provides an exhaustive evaluation of an ideal routing protocol. This values will allow a protocol designer to improve or fine tune his proposal. An activation window indicates when a node is considered *active* or *inactive*. A node is active, i.e., it can either send or receive messages, during certain time windows inside $[0, T]$.

We describe an algorithm, called *STP algorithm* (STPA), that based on the current position and state of the nodes determines factors like: how many complete messages get to the destination, which is the smallest amount of time required by a packet to get to the destination, which path followed each packet, and so on.

We show that the complexity of the algorithm is $O(\sigma^2)$, that is polynomial with respect to parameter σ; where σ corresponds to the sum of all the instants of time during which all nodes are active.

The rest of this paper is organized as follows. Section II describe the proposed network model. Section III defines the objectives and presents the details of the STP algorithm. Finally, some conclusion and the description of the future work are presented.

II. Model definition

We represent an ad hoc network as a set of n nodes, $W = \{v_i\}_{i=1}^{n}$, each referring to a device placed on the $\mathbb{R}^2$ plane. We define a coordinate function:

$$co_i : [0, T] \rightarrow \mathbb{R}^2, \quad co_i(t) = (x_i(t), y_i(t))$$

where $[0, T]$ is the total interval of time during which there is network activity. The coordinate function provides the position of node v_i at any instant of time $t \in [0, T]$.

Each node v_i can be either *active* or *inactive*. A node is active, i.e., it can either send or receive messages, during certain time windows inside $[0, T]$, where we will suppose that time takes integer values (discretized time). The set of time windows is defined as:

$$tw_i = \{[t^i_{2k-1}, t^i_{2k}]\}_{k=1}^{p_i}$$

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
Copyright © 2011 – 2012 ICST ISBN 978-963-9799-18-9
DOI 10.4108/ICST.WIOPT2008.3166

where $0 \leq t_s^i < t_r^i \leq T$ if $s < r$ with $s, r \in \{1, 2, \ldots, 2p_i\}$. Node v_i can only send messages at time t, where $t \in [t_{2k-1}^i, t_{2k}^i]$ for some $k \in \{1, \ldots, p_i\}$. Outside tw_i node v_i is *inactive* and it can neither send nor receive any message.

The transmission range of a node is a function of the node itself and the sending time t. We use a generic range function, $R_i(t)$, defined as:

$$R_i : [0, T] \rightarrow \mathbb{R}^+ \cup \{0\}$$

so that $R_i(t) = 0, \forall t \notin \bigcup_{k=1}^{p_i} \left[t_{2k-1}^i, t_{2k}^i\right]$. This function can depend on many factors, like the RF technology adopted, the surrounding environment, the antenna type, and so on. If node v_i sends a message at time t, and node v_j is inside v_i's range at that instant of time, that is, $\|co_i(t) - co_j(t)\| \leq R_i(t)$, the message will reach v_j at instant $t + t_{ij}(t)$.

In this paper we will us the term "packet" to indicate a message that requires a unit of time to leave a node. A packet is therefore the message unit and we will suppose that all messages consist of one or several packets and that at any instant of time a node can send at the most one packet.

Function $t_{ij}(t) \in \mathbb{Z}^+ \cup \{0\}$ indicates the units of time it takes for a message to move from v_i to v_j. The node v_j will receive this message only if active, that is only if $t + t_{ij}(t) \in [t_{2k-1}^j, t_{2k}^j]$ for some $k \in \{1, \ldots, p_j\}$. We consider that $t_{ij}(t)$ is small enough so that $co_j(t) \approx co_j(t + t_{ij}(t))$, i.e., that during that period of time v_j stays inside the range of v_i, if $||co_i(t) - co_j(t)|| \leq R_i(t)$. We are basically supposing that, since sending time are so small compared to the nodes mobility, vertices are practically static during each sending operation.

The sending of a message from v_i at time t and the reception at node v_j at instant $t + t_{ij}(t)$ if v_j is in the v_i range, implies a cost $c_{ij}(t) \geq 0$. We consider in this work that the cost coincides with the sending time $t_{ij}(t)$.

Once node v_j receives the message from v_i and determines that it has to forward it, it will do it at time $t + t_{ij}(t) + r_j$, where function $r_j \in \mathbb{Z}^+ \cup \{0\}$ represents the processing time at node v_j. The instant of time $t + t_{ij}(t) + r_j$ must belong to the same activation window of $t + t_{ij}(t)$ and of v_j, otherwise the message will get lost at v_j.

We will suppose that in the network the following circumstances are verified:

- If a node v_i receives more than one packet at the same instant of time, because of interferences all the packets will be lost in v_i.
- A node does not forward a packet originated by itself nor resends a packet whose destination is itself, nor resends a packet sent previously by itself. Basically a packet is resent by a node only if it comes from different paths, thus avoiding loops.
- To save energy, a node v_i resends a packet at most g_i times, ignoring all packets coming after the g_i-th resending operation.
- If a node v_j is about to send a locally generate packet at instant t, but at the instant $t - r_j$ receives one and only one packet, except in the three cases listed below, it will send the packet received at instant t and it will send its own packet in the following available instant t, delaying if it is required any other own packets. If at instant $t - r_j$ v_j receives more than one packet, as explained before these packets will be lost and therefore the node will send its own packet at instant t. The three exceptions are:
  - If the packet that receives at $t - r_j$ is directed to it;
  - if a packet was sent by v_j in a previous instant of time;
  - if the packet received at instant $t - r_j$ is not directed to v_j and it already forwarded it g_i times;

  in the above three cases it will send its own packet at instant t.

III. The STPA Algorithm

This section describes the details of the proposed algorithm. We first formally describe the problem, then we give the details of the algorithm, and finally we compute its complexity.

A. Problem statement

We suppose an ad hoc network made of n nodes, $W = \{v_i\}_{i=1}^n$. About this network we know all the information mentioned in Section II referring to an interval of time $[0, T]$. We suppose that during interval $[0, T]$ the subset of nodes $\{v_{i_j}\}_{j=1}^r$, where $r < n$ and $i_j < i_{j+1}$ $\forall j$, generate (i.e., send) messages. That is, node v_{i_j} send m_{i_j} messages $M_1^{i_j}, M_2^{i_j}, \ldots, M_{m_{i_j}}^{i_j}$ respectively to nodes $v_{1_{i_j}}, v_{2_{i_j}}, \ldots, v_{m_{ij_{i_j}}}$ which must not be two disjunct. Each message $M_k^{i_j}$ has a unique destination and it is made of $p_k^{i_j}$ packets sent at instants of time $t_1^{k_{i_j}}, t_2^{k_{i_j}}, \ldots, t_{p_k^{i_j}}^{k_{i_j}}$ where $0 \leq t_1^{k_{i_j}} < t_2^{k_{i_j}} < \ldots < t_{p_k^{i_j}}^{k_{i_j}} < T$. Packets related to different messages can be alternated, like for example, $t_1^{k_{i_j}} < t_1^{h_{i_j}} < t_2^{k_{i_j}}$ supposing $h \neq k$, where node v_{i_j} after sending the first packet of message $M_k^{i_j}$ send the first packet of message $M_h^{i_j}$ and afterward the second packet of message $M_k^{i_j}$.

The answers that we will give with our approach are all oriented to offer criteria to determine how effective is the routing protocol we are proposing with respect to an "ideal" protocol. More specifically we will obtain:

- how many complete messages get to the destination;
- how many complete in-order messages get to the destination;
- which packets will be delayed and for how long;
- at which instant of time will each successful packet get to its destination;
- which is the smallest amount of time required by a packet to get to the destination;
- which path was followed by each packet.

B. The algorithms details

By default all variables that are not initially set to a different value must be considered as initialized at 0.

Each vertex v_m^h that appears in the graph refers to node v_m at the instant of time h. Associated to each vertex v_m^h there is a 4-component vector $te_m^h = (\tilde{m}, \tilde{h}, \hat{m}, \tilde{h}')$. This vector must be interpreted as if node v_m at instant h forwards the packet sent for the first time by node $v_{\tilde{m}}$ at instant $\tilde{h}$, and whose destination is node $\hat{m}$; this packet was initially scheduled to be sent by $v_{\tilde{m}}$ at instant $\tilde{h}'$. Note that $\tilde{h} > \tilde{h}'$ means that the corresponding packet was sent with a delay of $\tilde{h} - \tilde{h}'$ units of time.

The vertexes v_m^h that generate packets, will sometime have a second associated vector, ste_m^h, similar to the previous one, and a third one, tte_m^h. These two vectors are used to determine which packet will be eventually sent.

Each vertex v_m^h will have another associated vector cam_m^h, with at most $n-1$ components (where n is the total number of nodes). The i-th component of cam_m^h will indicate the i-th node through which passed the packet that v_m will possibly send at instant h. This vector is used to avoid loops.

Obviously, at the beginning $cam_m^h = \emptyset$ for all m and h. The first element of cam_m^h, when created, indicates the node that generated the packet.

In the algorithm description we will use the symbol $\oplus$ to indicate the addition of a component to vector cam_m^h; e.g., $(2,3) \oplus 4 = (2,3,4)$. Also, $c \in cam_m^h$ will indicate that one of the components of cam_m^h is node c; therefore $m \in cam_m^h$ will alert about the creation of a loop.

Each vertex v_m^h will have associated a binary variable $etiq_m^h$ which will be set to 1 if node v_m receives 2 or more packets at instant $h - r_m$. This variable is associated only to vertexes that do not generate packets. Vertexes v_m^h that generate packets will have associated another variable rec_m^h which will be set to 0 if node v_m does not receive any packet at instant $h - r_m$, or to 1 if it receives a unique packet at instant $h - r_m$, or to 2 if it receives two or more packets at instant $h - r_m$ (interferences). All the $etiq$ and rec variables will be set to 0 at the beginning.

Using $c(v_m^h, v_{m'}^{h'})$ we will indicate the cost of arc $(v_m^h, v_{m'}^{h'})$ in G; in this work this will always coincide with $h' - h$, that is with $t_{mm'}(h) + r_{m'}$. By $d^+(v)$ we will understand, as normal in graph theory, the number of arcs that comes out from vertex v. Finally a set N will contain all those te_m^h vectors that correspond to packets that could not be sent since their sending instant were delayed to an instant at which the node is inactive and will remain inactive until T.

Basically the algorithm starts from an initial set Q with all the vertexes v_m^h so that v_m belongs to the subset of nodes that generate their own messages and h is the instant of time at which v_m wants to send one of its packets; therefore Q will initially have as much vertexes as many locally generated packets that the nodes want to send in the interval $[0, T]$.

At each iteration the algorithm searches in strict increasing time order a vertex v_a^b in Q, and it eliminates it from Q, looking for all nodes v_m and instants of time $b + t_{am}(b)$ where, given the network conditions, it is guaranteed that the message sent from v_a at instant b will get to v_m.

For the cases where this guarantee exists, the algorithm adds the vertexes $v_m^{b+t_{am}(b)+r_m}$ to V and Q with their respective

Algorithm 1: STPA Algorithm: *First Phase*

```
Initialization();
while Q ≠ ∅ do
    v_a^b := v_m^h so that v_m^h ∈ Q and h = min{l | v_s^l ∈ Q};
    Q ← Q ~ {v_a^b};
    if te_a^b(1) = a and te_a^b(2) = b then
        if (rec_a^b ≠ 1) or (rec_a^b = 1 and ste_a^b(3) = a) or
        (rec_a^b = 1 and ste_a^b(1) = a) or (rec_a^b = 1 and
        |{v_a^l ∈ V | l < b and te_a^l = ste_a^b}| = g_a) or
        (rec_a^b = 1 and a ∈ cam_a^b) then
            Forward();
        else
            tte_a^b = te_a^b and te_a^b = ste_a^b ; Forward();
            delay the sending of v_a at b' > b, where b' is
            the first avail. inst. that v_a is active at b';
            if v_a^b' ∉ Q then
                Q ← Q ∪ {v_a^b'}; V ← V ∪ {v_a^b'};
                rec_a^b' = 0 and
                te_a^b' = (a, b', tte_a^b(3), tte_a^b(4));
            if v_a^b' ∈ Q then
                if (te_a^b'(1) ≠ a) or (te_a^b'(1) = a and
                te_a^b'(2) < b') then
                    if etiq_a^b' = 1 then
                        rec_a^b' = 2, ste_a^b' = te_a^b',
                        te_a^b' = (a, b', tte_a^b(3), tte_a^b(4));
                        A ← A ~
                        {(u, v_a^b') | (u, v_a^b') ∈ A} and
                        cam_a^b' = ∅;
                    if etiq_a^b' = 0 then
                        rec_a^b' = 1, ste_a^b' = te_a^b';
                        te_a^b' = (a, b', tte_a^b(3), tte_a^b(4));
                if te_a^b'(1) = a and te_a^b'(2) = b' then
                    tte_a^b' = te_a^b' and
                    te_a^b' = (a, b', tte_a^b(3), tte_a^b(4));
                    Delay(v_a)
    if (te_a^b(1) ≠ a) or (te_a^b(1) = a and te_a^b(2) < b) then
        if (etiq_a^b = 1) or (etiq_a^b = 0 and te_a^b(1) = a) then
            V ← V ~ {v_a^b};
            A ← A ~
            {(u, v_a^b) | (u, v_m^{b+t_am(b)+r_m}) ∈ A};
        if (etiq_a^b ≠ 1 and te_a^b(1) ≠ a) then
            if te_a^b(3) ≠ a then
                if |{v_a^l ∈ V | l < b and te_a^l = te_a^b}| = g_a
                or a ∈ cam_a^b then
                    V ← V ~ {v_a^b}; A ← A ~ {(u, v_a^b)}
                    being (u, v_a^b) the unique arc entering
                    in v_a^b;
                else
                    Forward();
SecondPhase();
```

Procedure `Initialization`

begin

$Q := \{v_m^h \mid$ so that v_m belongs to the subset of nodes that generate local messages and h is the instant of time when v_m should send one of its packets$\}$;

$V := Q$;

$A := \emptyset$;

$te_m^h = (m, h, \tilde{m}, h)\ \forall v_m^h \in Q$, where $v_{\tilde{m}}$ is the destination node of the packet that v_m wants to send at instant h;

end

vectors $te_m^{b+t_{am}(b)+r_m}$, and the arcs $(v_a^b, v_m^{b+t_{am}(b)+r_m})$ to A, or it properly mark them if they already belong to Q, depending on node v_m receives at instant $b + t_{am}(b)$ two or more packets and therefore, due to interferences, it loses the information of those packets or it generate a local packet at instant $b + t_{am}(b) + r_m$.

If the third vector component of the selected vertex v_a^b is a (i.e., $te_a^b(3) = a$), means that v_a is the destination of the received packet, then v_a^b will be eliminated from Q without any further search, unless v_a generates at the same instant of time a local packet at $t = b$. Iteration stops when $Q = \emptyset$.

Finally, to reduce as much as possible the size of G we eliminate from V in strict decreasing time order all those vertexes v_m^h without leaving arcs ($d^+(v_m^h) = 0$) where v_m does not generate any local packet at instant $t = h$ and $te_m^h(3) \neq m$, which means that v_m is not the destination of the received packet at instant $h - r_m$, eliminating at the same time its input arc of A, which can generate more vertexes without any leaving arc in G at previous instant. The resulting directed graph $G = (V, A)$ will be acyclic and will be formed by:

- Maximum paths, they are not part of a longer path, which go from vertexes v_m^h so that v_m belongs to the subset of nodes the generate their own messages and h is the instant of time at which v_m send one of its packets, to vertexes $v_{m'}^{h'}$ so that $v_{m'}$ is the destination node of the packet originally send by v_m at $t = h$.
- Various of the previous paths concatenated among them, if it happens that $v_{m'}^{h'}$ is at the same time generator of a local packet.
- Isolated vertexes that correspond to packets sent which did not get to their final destination.

Two paths corresponding to two different packets will be vertex-disjoint, except in the case that they are concatenated. In this last case they will have in common the last vertex of the former and the first vertex of the latter.

Thanks to how the algorithm is designed and to some basic properties of graph theory it is easy to demonstrate the following statements, which we have included in the following unique theorem:

Theorem 1: Given an ad hoc network, described with all the parameters of Section II and given the graph G and the set of vectors N obtained applying the STPA algorithm, it is verified that:

a) A packet generated and sent by v_s at instant t will not get to its destination if and only if $v_s^t \in V$ and $d^+(v_s^t) = 0$.

b) A packet that was scheduled to be generated and sent by v_s at instant t to v_d, will be definitively sent by v_s at instant t', where $t' \geq t$ if $v_s^{t'} \in V$ and $te_s^{t'} = (v_s, t', v_d, t)$.

c) A packet that was scheduled to be generated and sent

Procedure `Forward`

begin

forall v_m *with* $m \neq a$, $\|co_a(b) - co_m(b)\| \leq R_a(b)$, v_m *active during the interval* $[b + t_{am}(b), b + t_{am}(b) + r_m]$ *and* $b + t_{am}(b) + r_m \leq T$ **do**

  if $v_m^{b+t_{am}(b)+r_m} \in Q$ **then**

    if *(*$te_m^{b+t_{am}(b)+r_m}(1) \neq m$*) or* *(*$te_m^{b+t_{am}(b)+r_m}(1) = m$ *and* $te_m^{b+t_{am}(b)+r_m}(2) < b + t_{am}(b) + r_m$*)* **then**

      $A \leftarrow A \cup \left\{(v_a^b, v_m^{b+t_{am}(b)+r_m})\right\}$;

      $etiq_m^{b+t_{am}(b)+r_m} = 1$;

    if $te_m^{b+t_{am}(b)+r_m}(1) = m$ *and* $te_m^{b+t_{am}(b)+r_m}(2) = b + t_{am}(b) + r_m$ **then**

      if $rec_m^{b+t_{am}(b)+r_m} = 0$ **then**

        $ste_m^{b+t_{am}(b)+r_m} = te_a^b$;

        $rec_m^{b+t_{am}(b)+r_m} = 1$;

        $A \leftarrow A \cup \left\{(v_a^b, v_m^{b+t_{am}(b)+r_m})\right\}$;

        $cam_m^{b+t_{am}(b)+r_m} = cam_a^b \oplus a$;

        $c(v_a^b, v_m^{b+t_{am}(b)+r_m}) := t_{am}(b) + r_m$;

      if $rec_m^{b+t_{am}(b)+r_m} = 1$ **then**

        $rec_m^{b+t_{am}(b)+r_m} = 2$,

        $cam_m^{b+t_{am}(b)+r_m} = \emptyset$;

        $A \leftarrow A \sim \left\{(u, v_m^{b+t_{am}(b)+r_m})\right\}$ being $(u, v_m^{b+t_{am}(b)+r_m})$ the unique arc entering in $v_m^{b+t_{am}(b)+r_m}$;

      if $rec_m^{b+t_{am}(b)+r_m} = 2$ **then**

        do nothing

  if $v_m^{b+t_{am}(b)+r_m} \notin Q$ **then**

    $Q \leftarrow Q \cup \left\{v_m^{b+t_{am}(b)+r_m}\right\}$;

    $V \leftarrow V \cup \left\{v_m^{b+t_{am}(b)+r_m}\right\}$;

    $A \leftarrow A \cup \left\{(v_a^b, v_m^{b+t_{am}(b)+r_m})\right\}$;

    $cam_m^{b+t_{am}(b)+r_m} = cam_a^b \oplus a$;

    $c(v_a^b, v_m^{b+t_{am}(b)+r_m}) := t_{am}(b) + r_m$;

    $te_m^{b+t_{am}(b)+r_m} = te_a^b$;

end

Procedure `Delay(`v_a`)`

begin

Delay the sending of the local packet of v_a forecasted for instant b' to the next instant $b'' > b'$ so that v_a is active at b'', doing exactly as with v_a^b, changing b for b' and b' for b", and so successively until there is no need to delay any local packet of v_a that was to be sent after b;
If, as a consequence of this process, a packet that was to be sent from v_a, cannot be sent because delayed too long and v_a becomes inactive; the vector corresponding to this packet must be stored in N;

end

Procedure `SecondPhase`

begin

$P := \{v_m^h \in V \mid d^+(v_m^h) = 0,$
$te_m^h(3) \neq m \ and \ te_m^h(1) \neq m\}$;
while $P \neq \emptyset$ **do**
$v_a^b := v_m^h$ so that $h = \max\{l \mid v_m^l \in P\}$;
$P \leftarrow P \sim \{v_a^b\}$;
$V \leftarrow V \sim \{v_a^b\}$;
$A \leftarrow A \sim \{(u, v_a^b)\}$ being (u, v_a^b) the unique arc entering in v_a^b;
update P;

end

by v_s at instant t to v_d, will not be sent by v_s at any instant of time if a vector $(v_s, t', v_d, t) \in N$ for some $t' \geq t$.

d) If v_s generates and sends a packet at instant t to v_d, this packet will get to its destination at instant t' if it exists in G a path from v_s^t to $v_d^{t'+r_d}$.

e) Given a maximum path in G that starts from v_s^t and ends in $v_d^{t'}$, then either $te_s^t(3) = v_d$ or two vertexes exist inside that path in G, v_i^h and $v_j^{h'}$ with $t < h \leq h' < t'$ (possibly $i = j$ and $h = h'$) so that $te_s^t(3) = v_i$ and $te_j^{h'}(3) = v_d$.

Thanks to the information stored into vectors te_d^t we know which, if any, packet gets at each instant to v_d. For example, if we want to know at which instant the packet sent from v_s at instant m will get to v_d, we will have to look for the vertex v_d^h in V, if it exists, so that $h = \min\{l \mid v_d^l \in V$ and $te_d^l = (s, m, d, m')\}$. This packet will get to v_d at instant $h - r_d$. The time required by the packet since it left v_s at instant m will be $h - r_d - m$ time units. It was initially scheduled to be sent at m', where $m > m'$, meaning that the packet suffered a delay of $m - m'$ time units. The information about the path followed by the packet is stored in cam_d^h. By applying a simple method of graph theory to compute the unique path from v_s^m to v_d^h in G and checking the super-indexes of the vertexes in the path we can determine at which instant of time the packet has been forwarded by each node in the path.

Item *e* of the previous theorem means that given a maximum path in G that starts from v_s^t and ends in $v_d^{t'}$, two situations can occur: either v_s generates and send a packet at instant t to v_d, which will receive it at $t' - r_d$, or this maximum path is made of various concatenated paths, as if for example node v_i receives a packet being the final destination at instant $h - r_i$ and at the same time it generates and send a packet at instant h (vertex v_i^h inside the path).

C. Complexity computation of STPA

In this section we determine the upper bound for the complexity of the STPA algorithm. We will not minimize this upper bound but we will show that the complexity of the algorithm is polynomial with respect to parameter σ. This parameter corresponds to the sum of all the instants of time during which all nodes are active, that is:

$$\sigma = \sum_{i=1}^{n} \sum_{j=1}^{p_i} (t_{2j}^i - t_{2j-1}^i)$$

If we define $\sigma_i = \sum_{j=1}^{p_i} (t_{2j}^i - t_{2j-1}^i)$ where $i \in \{1, 2, \ldots, n\}$, we can redefine the previous definition of σ as: $\sigma = n\bar{\sigma}$ where $\bar{\sigma}$ is the average value of all σ_i, that is of the instants of time at which each node is active in $[0, T]$.

Theorem 2: The STPA algorithm has complexity $O(\sigma^2)$.

Proof:

The number of iterations made by the algorithm in the first phase is bounded above by σ since at each iteration it sets a different v_s^t and it eliminates it from Q. At each iteration, the number of comparisons required to look for the selected vertex is also bounded above by σ, since $|Q| \leq \sigma$.

Once v_a^b is selected, the procedures that impact the complexity of the algorithm are:

- Compute $|\{v_a^l \in V \mid l < b$ and $te_a^l = ste_a^b\}|$, with a number of basic operation $O(\sigma_a)$ (we have to check a number of instants l bounded by σ_a).
- The process of, if it is the case, delaying the sending of a locally generated packet, requires also $O(\sigma_a)$ basic operations. These operations will be done at an instant of time after b when v_a is active.
- The subroutine FORWARD requires just a few operations for each node v_m where $m \neq a$, basically checking if it is active over the interval $[b + t_{am}(b), b + t_{am}(b) + r_m]$. The value r_m should be very small, we consider the bound for this interval width equal to σ_m, and therefore the complexity of this subroutine is $O(\sigma)$.

Regarding the second phase, a few checks must be done for each vertex V, where $|V| \leq \sigma$ to determine P. P will have less vertexes than $|V|$, vertexes which will require just a few basic operations. Therefore the complexity of this second phase is $O(\sigma)$ and the total complexity of the algorithm is $O(\sigma^2)$.

■

IV. Conclusions

In this paper we presented an algorithm called STPA (Shortest Time Path Algorithm) which aims at providing a comparison tool for the design of routing algorithm for Mobile Ad Hoc Networks (MANETs).

Instead of validating new proposals by using simulation with typical tools like ns2, what STPA provides is an exhaustive evaluation of an ideal routing protocol. Based on the current position and state of the nodes it can determine factors like: how many complete messages get to the destination, which is the smallest amount of time required by a packet to get to the destination, which path followed each packet, and so on. This values would allow a protocol designer to improve or fine tune his proposal.

We demonstrate that the complexity of the algorithm is $O(\sigma^2)$, i.e., polynomial with respect to parameter σ; where σ corresponds to the sum of all the instants of time during which all nodes are active.

This is anyway a preliminary work. We are currently designing a software tool that will exhaustively evaluate a given scenario and that will provide all the characterizing parameters of an ideal routing protocol.

Mobility will also be introduced to completely evaluate MANETs scenarios.

V. Acknowledgments

This work was partially supported by FEDER and the *Ministerio de Educación y Ciencia*, Spain, under Grant TIN2005-07705-C02-01.

References

[1] Juan Carlos Cano and Pietro Manzoni. Evaluating the energy-consumption reduction in a MANET by dynamically switching-off network interfaces. *Proceedings of the 6th IEEE Symposium on Computers and Communications, Hammamet, Tunisia*, July 2001.

[2] Hao Yang, James Shu, Xiaoqiao Meng, and Songwu Lu. Scan: Self-organized network-layer security in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):261–273, 2006.

[3] K. Fall and K. Varadhan. ns notes and documents. The VINT Project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, February 2000. Available at http://www.isi.edu/nsnam/ns/ns-documentation.html.

[4] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes fairness in distributed ad-hoc networks. In *Proceedings of IEEE/ACM MobiHOC, 2002.*

[5] A. Faragó and V. R. Syrotiuk. Merit: a scalable approach for protocol assessment. *Mob. Netw. Appl.*, 8(5):567–577, 2003.

[6] Jing Tian, Joerg Haehner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *35th Annual Simulation Symposium, April 14 - 18, 2002, San Diego, California*.