Space-Time Routing in Ad Hoc Networks *

Henri Dubois-Ferrière[1], Matthias Grossglauser[1], and Martin Vetterli[1 2]

[1] School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland
[2] Department of EECS, University of California Berkeley, USA
Henri.Dubois-Ferriere@epfl.ch, Matthias.Grossglauser@epfl.ch,
Martin.Vetterli@epfl.ch

Abstract. We introduce Space-Time Routing (STR), a new approach to routing in mobile ad hoc networks. In STR, the age of routing state is considered jointly with the distance to the destination. We give a general description of STR, which can accommodate various temporal (age) and spatial (distance) metrics. Our formulation of STR describes a family of routing algorithms, parameterized by a choice of node clock scheme, a neighbor-distance function and a binding spatio-temporal metric which allows the algorithm to compare potential routes taking into account both their age and their distance to the destination. We discuss possible instantiations of a Space-Time Routing protocol. In particular, we review FRESH (FResher Encounter SearcH), a routing algorithm using temporal information only, and GREP (Generalized Route Establishment Protocol), a routing protocol which uses jointly spatial and temporal information about routes. We discuss a third STR algorithm using only physical notions of space and time, and finally show that STR provides loop-free routes.

1 Introduction

An ad hoc network is a communication medium where users or nodes also provide the infrastructure for communication. That is, nodes play both the role of terminals (i.e. source and destination of messages) and of relays. Thus, a message traverses an ad hoc network by being relayed from node to node, until it reaches its destination. When, in addition, nodes are moving, this becomes a challenging task, since the topology of the network is in constant flux. How to find a destination, how to route to that destination, and how to insure robust communication in the face of constant topology change are major challenges in mobile ad hoc networks.

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322

Routing in ad hoc networks is a well studied topic, with a number of proposed protocols like AODV [1] and DSR [2], as well as simulation studies. A common point of existing algorithms is that their computations involve almost exclusively *distance* (or *spatial*) types of information. This approach can be traced all the way back to the classic Dijkstra, Bellman-Ford, and Floyd-Warshall algorithms, which are driven by quantities measuring *distances*[3] between nodes.

However these spatial routing algorithms were designed with an assumption of static or near-static topologies, where nodes do not move and links change at a slow rate (if at all). In previous work [3], we considered the situation where all nodes are constantly moving, making therefore topology change the norm rather than the exception. In such a scenario, we showed that a routing algorithm that was driven *exclusively by temporal metrics* could significantly outperform spatial approaches. Specifically, we introduced an algorithm named FRESH (FResher Encounter SearcH). Using a simple flood-based search primitive, FRESH advances toward the destination by searching iteratively for a node which has encountered the destination *more recently* than the current node.

FRESH took the extreme approach of using only temporal information in order to demonstrate the value of such information for routing in highly mobile ad hoc networks. However it is clear that spatial information can still be useful, and that ignoring spatial state that exists in the network is highly suboptimal. Now, given that temporal information can increase routing efficiency, and that spatial information remains useful, the question is: Are spatial and temporal approaches incompatible and distinct, or can we design routing algorithms which incorporate seamlessly both aspects?

The purpose of this paper is to answer the above question by introducing a unifying view of routing in highly mobile networks using jointly *both temporal and spatial* information. We call such an approach *Space-Time Routing* (STR).

The central intuition underlying STR is the following. When the rate of topology change increases, the average time during which spatial information remains exact is reduced. For example, a routing entry saying that the destination is reachable from node S in 8 hops through neighbor N becomes inexact if N moves, or if intermediate nodes move such that the number of hops is different than 8. However, even if the routing entry is not perfectly accurate anymore, it can still be helpful. In other words: *aged, inexact routing state is valuable, and incorporating temporal information about the age of routes allows the algorithm to make full use of all available information, including partially outdated routes.* This can be contrasted with spatial-only approaches which are predicated on routing state being exact (since they have no way of 'weighing' the accuracy of aged state). As a result, when spatial algorithms (conceived for mostly-static graphs) are transposed to mobile ad hoc routing protocols, the protocols must be very aggressive in timing out state in order to avoid as far as possible having

[3] equivalently, *transmission costs.*

outdated routes which the protocols are not equipped to handle. For example the default route timeout in AODV [1] is 3 seconds.

Just as spatial routing algorithms can use different distance metrics, STR is amenable to various spatial, temporal, and joint spatio-temporal metrics. Specifically, a particular STR algorithm is defined by the choice of

- physical or logical notion of time,
- a spatial neighbor-distance function $\triangle$, and
- a binding spatio-temporal (S-T) metric f.

Therefore we provide a general formulation of STR which is independent of the specific metric choices. The neighbor-distance metric can be logical (e.g., number of hops) or physical (e.g., euclidean distance, energy cost). The binding S-T metric is used to compare two route entries to a destination of different distance and age and to decide which is closest in the joint spatio-temporal space.

The rest of the paper is organized as follows. In Section 2, we give a general formulation of STR and discuss some properties. In Section 3, we give examples of two specific STR algorithms: FRESH, GREP, and outline a third algorithm using physical notions of space and time. In Section 4, we discuss some properties of STR, including loop-freedom. Section 5 concludes the paper.

2 Space-Time Routing

2.1 Notation and Assumptions

We note $V = \{1 \ldots n\}$ the set of nodes in the network, and E the set of edges $(i, j) \in E$ for $i, j \in V$. Associated with the set of edges is a distance function[4] $\triangle : E \rightarrow \mathbf{R}$. For example if distance is counted as the number of hops, we would have $\triangle(i, j) = 1$. We assume that any node can obtain the distance to its neighbors (trivially in the case of hop-count distance, or for example using a signal-strength based estimation in the case of euclidean distances). Each node maintains its own clock, which is used to stamp every packet with the clock time of the node which originates it. Simple examples of a node clock are a *physical* (oscillator-based) clock giving a continuous reading, for example in seconds, or a *logical* clock providing a discrete ordering of routing events relative to that source. Whichever temporal representation is used, STR does not require any form of inter-node clock synchronization.

[4] Note that given node mobility, E and $\triangle$ vary over time. For simplicity of notation we drop the time index since we only refer to the values of E and $\triangle$ "at the present time".

Then STR requires a *binding spatio-temporal metric*, which is a function $f : \mathbf{R}^2 \rightarrow \mathbf{R}$, taking as input a (spatial) distance value and a (temporal) clock value and returning a scalar representing the "norm" of this pair in the spatio-temporal space. The binding S-T metric must satisfy the following conditions:

$$argmin f(s,t) = (0,0) \tag{1}$$

For fixed d, f is an increasing function of t

$$sgn(f(d,t_1) - f(d,t_2)) = sgn(t_1 - t_2) \tag{2}$$

For fixed t, f is an increasing function of d

$$sgn(f(d_1,t) - f(d_2,t)) = sgn(d_1 - d_2) \tag{3}$$

Routing Table Entries. Each node maintains a distance-vector routing table containing one entry for each destination node. In addition to the *next hop* and *distance* fields which are used in spatial routing algorithms, STR routing entries also include the *age* of the entry.

Table 1. Routing table entries

n_D^N	Next hop to node D in N's routing table.
d_D^N	Distance to node D in N's routing table.
t_D^N	Source Time of the routing entry to D in N's routing table.

Table 1 summarizes the notation used to describe routing state at each node. We drop the superscript and use the notation n_D, d_D, t_D when the context allows doing this unambguously. We use the convention that when a node has no entry for D, $n_D = null$, $d_D = \infty$, and $t_D = \infty$.

Packet Types. Beside regular data packets, STR uses *route request* (RREQ) and *route reply* (RREP) packets. A node sends a RREQ packet when it has no route to the destination, or if the next hop along the route is broken. It sends a RREP packet in reply to a route request when it has a route fresh enough and short enough to satisfy that request. Note that STR does not use any route error packets: since link breaks are always repaired locally, there is no need to inform the source and upstream nodes when this occurs.

Apart from the usual source and destination addresses of a packet, we introduce the following STR-specific fields: The *source time* of a packet ($p.st$) is the clock time at the packet's source node when it originated the packet. Each packet is stamped with the clock time of the source that originates it. This field is present in all packets.
The *source distance* of a packet ($p.sd$) is the distance this packet has traversed

since leaving the source. It is updated at each hop to reflect the new distance from the source. This field is present in all packets.
The *destination distance* ($p.dd$) and *destination time* ($p.dt$) of a packet are present only in RREQ and RREP packets. In the case of a RREQ packet, they come from the routing entry that the source of the RREQ has to the requested destination. In the case of a RREP packet, they represent the distance to the requested destination from the replying node.

2.2 STR Algorithm

DATA Processing. We first describe originating and forwarding of data packets. A node S originating a data packet initializes the source distance field to 0 and initializes the source time field to the present value of its clock. A node N receiving from neighbor M a packet originated by node S first increments the source distance field of the packet to reflect the distance that the packet has now traversed: $p.sd \leftarrow p.sd + \triangle(N, M)$.
Then, if the packet has come over a shorter (in the spatio-temporal metric space) route than the route it currently has, N updates its routing entry to S. Formally, if $f(p.sd, p.st) < f(d_S, t_S)$, then N updates its routing entry for S as:
$d_S \leftarrow p.sd \qquad t_S \leftarrow p.st \qquad n_S \leftarrow M$.
If the destination D of the packet is N itself, then no further processing is needed. If the destination is another node, N forwards the packet to its next hop n_D. If $n_D = null$, or if forwarding fails, N buffers the packet and initiates a route request procedure.

RREQ Processing. A node N initiating a route request procedure for destination D sets the source distance and source time fields as for a DATA packet. The destination distance and destination time fields on the packet are set respectively with the values d_D and t_D from N's routing table (with a suitable encoding when $d_D = \infty$ and $t_D = \infty$).
A node N receiving from neighbor M a RREQ packet originated by S increments the source distance of the packet and (possibly) updates its routing entry for S according to the same procedure as for a DATA packet. N then verifies if the spatio-temporal distance of its routing entry to D is smaller than the sum of S's spatio-temporal distance to D and the distance traveled by the packet, and originates a RREP to M if this is true.
Formally, if $f(d_D, t_D) < f(p.dd, p.dt) + f(p.sd, 0)$, then N initiates a RREP packet to S. Otherwise N re-broadcasts the RREQ packet (RREQ floods will be scoped with a time-to-live (TTL) field; we omit the details).

RREP Processing. A node R originating a RREP packet sets the source distance and source time fields as for a DATA packet. The destination distance and destination time fields are set respectively with the values d_D and t_D from N's routing table.
We consider now a node N receiving from neighbor M a RREP packet originated

by R, in response to a route request for a route to node O (if $R = O$ then the route reply was initiated by the destination itself, otherwise we say that R sent a route reply *on-behalf-of* O. N first increments the source distance of the packet and (possibly) updates its routing entry for R following the same procedure as for a DATA packet. Then, N updates its routing entry to O, if this will result in a shorter route (in the spatio-temporal metric space).
Formally, if $f(d_O, t_O) < f(p.dd, p.dt) + f(p.sd, 0)$, then N updates its routing table as: $d_O \leftarrow p.sd + p.dd \qquad t_O \leftarrow p.st \qquad n_O \leftarrow M$.
Then, if N has updated its routing entry to O, it forwards the RREP packet toward the originator of the route request (as determined by the destination field in the RREP packet). Otherwise N silently discards the RREP packet. When the RREP arrives at the node which originated the route request, this node can now forward its buffered DATA packets along the newly established route.

2.3 Discussion

Protocol-Specific Optimizations. The above exposition of STR is voluntarily simple and does not include possible optimizations that would be present in a full, practical protocol. As a first example of optimizations that might be present in a full protocol specification, a node receiving (or overhearing) any packet from a neighbor can update its routing entry to have a current, one-hop route to that neighbor. A second example pertains to route requests which in practice will be scoped using a TTL mechanism, and will likely proceed according to an expanding ring search. Expanding ring searches are used in many ad hoc routing protocols; the specifics of this procedure are omitted here. We refer to [4] as an example of a complete, practical STR protocol formulation.

A final example relates to proactive operation of STR. The formulation given here is *purely reactive*, meaning that a route is only computed when it is required to send packets. However STR can also accommodate proactive, or hybrid proactive/reactive operation, whereby nodes proactively inform other nodes of some or all of routing entries. This is done using *route advertisement* packets, and a route update decision mechanism similar to that used when receiving a regular packet: if an advertised route is shorter in the S-T space than the one in the receiving node's routing table, then it overrides the existing route. We refer to [4] for a simple example of route advertisement operation in a STR protocol. Many schemes for controlling the proactive dissemination of routing information are possible. For example, [5] explore schemes to adjust the relative amount of reactive and proactive overhead. With STR another possibility would be to define a threshold value ω such that a node N proactively disseminates only the routing entries which satisfy $f(d_D, t_D) < \omega$.

On Explicit versus Implicit Use of f. We remark that in this exposition of STR, the actual value of the spatio-temporal distance metric need not be explicitly computed. Specifically, the STR algorithm only uses the S-T metric

to compare two values, in order to decide if $f(d_1, t_1)$ is smaller or greater than $f(d_2, t_2)$. This has two consequences. The first is that two S-T metrics will result in identical STR protocols, if they induce the same ordering. For example all functions in the one-parameter family $f_k(d,t) = k(d+t)$ will result in the same ordering for any choice of $k > 0$. The second consequence is that the function f need not be explicitly defined. For example, in the case of the GREP protocol (see Sect. 3), the 'natural' presentation does not define f explicitly, though of course a function f resulting in the equivalent ordering can be defined.

3 Three Instances of STR

We now give three instances of specific STR algorithms which provide a sample of the wide range of protocols that fit under the STR umbrella.

3.1 FRESH: FResher Encounter SearcH

FRESH [3] is a simple route discovery algorithm using exclusively temporal information. Nodes keep a record of their most recent encounter times with other nodes. Instead of searching for the destination, the source node searches for any intermediate node that encountered the destination *more recently than did the source node itself.* The intermediate node then searches for a node that encountered the destination yet more recently, and the procedure iterates until the destination is reached. Therefore, FRESH replaces the single network-wide search of current proposals with a succession of smaller searches, resulting in a cheaper route discovery.

The formulation originally employed in [3] was a direct algorithmic transposition of the above description, using at each iteration of an underlying *search primitive* which roughly corresponded to the flooding and reverse-path setup phase of STR.
We now show that FRESH can be expressed as a STR algorithm. First, FRESH uses physical time, so packets are stamped with the clock time of the node which originates them. Second, the spatial distance is measured in hops: $\triangle_{FRESH}(i,j) = 1$. Finally, the binding S-T metric ignores all spatial information and only compares one-hop encounter times:

$$f_{FRESH}(d,t) = \begin{cases} \infty \text{ if } d > 1; \\ t \quad \text{if } d \leq 1. \end{cases}$$

We note that the function f_{FRESH} alone, when inserted into the STR description of Sect. 2 does not result in the exact FRESH algorithm [3]. This would require distinguishing in STR two binding spatio-temporal metrics, one of which (corresponding to this f_{FRESH}) to be used in deciding whether a node's route is suitable to answer a route request, the other to be used in deciding whether to update the reverse-path entry to the source of an incoming packet.

3.2 GREP: Generalized Route Establishment Protocol

GREP [4] is a complete, practical routing protocol which demonstrated that a protocol incorporating both spatial and temporal metrics was not only feasible but also highly efficient compared to spatial-only approaches. Though the original proposal for GREP predates the general formulation of STR given in this paper, we now show how GREP can be defined as an instance of STR.

First, GREP uses logical clocks, similar to Lamport's clocks [6]. Each node maintains its own integer-valued clock, and increments it each time it transmits a packet. Packets are stamped with the logical clock time of the node which originates them, and are therefore similar to *sequence numbers* as used in many routing protocols [1]. As in FRESH, neighbor distances are measured in hops: $\triangle_{GREP}(i,j) = 1$.

In the original proposal for GREP, the binding S-T metric was not explicitly computed. Rather, the ordering between two (s,t) pairs was obtained as:

$(s_1,t_1) < (s_2,t_2)$ if $t_1 < t_2$ or $((t_1 = t_2)$ and $s_1 < s_2)$.

We believe that the above formulation is the most expressive for GREP since it captures the notion that GREP advances (in spatial mode) along a route segment of particular age, then switches to temporal mode if the next hop along this route is broken, and looks for any route of younger age. However, as with any STR algorithm, it is possible to explicitly define the binding S-T metric of GREP:

$$f_{GREP}(d,t) = d + k * t$$

where k is a suitably large constant (for example $k = |V|$).

3.3 STR with physical space and time

Our third example of STR is based on a physical representation of both spatial and temporal distances, and illustrates how a priori knowledge of the mobility process may be exploited in designing the binding metric f.

We note X_n the euclidean position of node n. Now $\triangle$ measures euclidean distance between neighbors: $\triangle(i,j) = \|X_i - X_j\|$. Node clocks measure physical time as for FRESH.

The binding metric in this case has the form

$$f(d,t) = d + cvt^{\alpha}.$$

Note that the unit of v here is [m/s]. One possible choice would be to set v to the average velocity of nodes, in order to have the S-T metric reflect a quantity

related to the expected present distance to a particular node. A suitable choice for the parameter α would depend on the mobility process assumed. For example, in a waypoint model, nodes traverse a distance which is proportional to time elapsed, which would indicate the choice $\alpha = 1$. Or with a random walk, one may choose $\alpha = 1/2$, since the time taken to traverse a distance d is $O(d^2)$.

4 Analysis and Properties

Loop Freedom. In this section we show that STR is free of routing loops. We distinguish between *packet loops* and *route loops*. A packet loop happens when a unicast packet traverses the same node twice. A route loop happens when a unicast packet traverses the same node twice, and the routing state pertaining to the packet's destination at that node does not change between both traversals. A route loop is potentially infinite (unless some mechanism is used to kill packets which have traversed more than some number of hops). In other words a packet gets "stuck" in a route loop but not in a packet loop, since the routing state has changed when it traverses the same node for the second time.

The route loop-free operation of STR comes from a simple observation: *At each hop, a packet advances to a node which is closer to the destination in the spatio-temporal metric space.*. This can be stated equivalently in terms of node routing tables:

Lemma 1. *If* $n_D^N = M$ *then* $f(d_D^M, t_D^M) < f(d_D^N, t_D^N)$.

The proof of this lemma follows from the fact that at each step in the protocol operation, (see Sect. 2) a new routing entry can only override an existing one if it offers a shorter $f(d,t)$ S-T distance to the destination.
We can now show that STR is route loop-free:

Theorem 1. *A packet routed by STR can never enter a route loop.*

Proof. Let us assume that a packet p is in a route loop, that is that it traverses node N twice with t_D^N, n_D^N, and d_D^N having identical values at both times. This is immediately contradicted by Lemma 1 since at each hop the packet advances in the spatio-temporal metric space.

On Packet Loops. We have shown above that STR routes are loop-free. Our analysis has made the distinction between *packet loops* and *route loops*. This distinction is usually not made in the analysis of routing protocols because they often prove loop freedom by showing that a packet will not traverse the same node twice; therefore both packet and route loops are excluded.

STR, on the other hand, excludes route loops but does not exclude packet loops. Therefore it offers a weaker guarantee than protocols such as [1] [2] which establish routes on an end-to-end basis and require a route to be converged before sending packets. This weakened guarantee can been seen as a consequence of STR's distributed hop-by-hop operation which uses only local repairs without involving the end-points of a route. On the other hand, relaxing protocol guarantees to allow packet loops allows an increase in efficiency which make this particularly worthwhile in highly mobile networks.

We discuss a small example of a packet loop showing that even when a packet loop does occur, subsequent packets will shortcut the loop and therefore packet loops cannot happen on back-to-back packets. In Fig. 4, there is a route from S to D, which might have been established by a packet sent earlier from D to S with sequence number 1. D has since moved and therefore the last hop of this route is broken.

This example shows an instance of a packet loop since the data packet traverses node B twice. Note that this is not a *route loop* since B's routing entry for destination D has changed between the first and second traversals, and therefore the packet does not get "stuck" in a loop between B and C. Note also that subsequent packets for D will now be forwarded by B to E; *each instance of a packet loop can only occur once.*

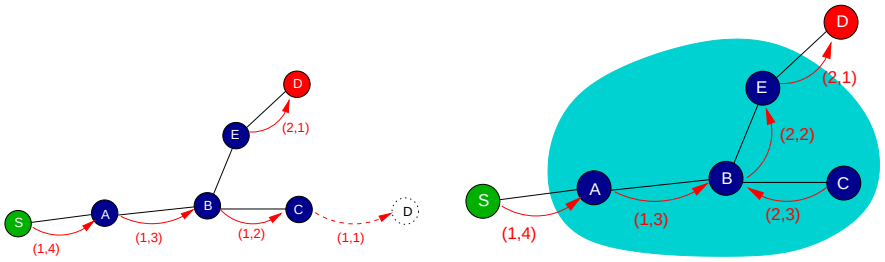

Fig. 1. On the left side: A network with a route from S to D having sequence number 1. D has moved, breaking the last hop. A packet sent by S to D is buffered at C while C sends a route request. On the right side: C's route request is answered by D, resulting in a new route with sequence number 2. The packet for D buffered at C can now be forwarded back through B, resulting in a packet loop ($S - A - B - C - B - E - D$). A packet loop only occurs once; subsequent packets from S to D will be routed by B directly to E.

Exploiting Outdated Routing State. Most ad hoc routing protocols attach some notion of *useful lifetime* to their routing state. Typically each route (or individual routing entry) is expired when it remains unused past a certain timeout (3 seconds in AODV for example).

In GREP *routing state never times out*: a routing entry can only be deleted when a newer entry overrides it. This is because a past route, which was often acquired at a high flooding cost, can still carry noisy, but useful information about the present topology.

We consider two simple scenarios to illustrate this. The first scenario is straightforward and concerns a short-term timescale. Consider a route which has been left unused for some time. In this time, one of the nodes in the route has moved. Clearly, timing out the whole route at this point would impose a costly rediscovery if the route is needed again; if we keep all routing entries only a small, local repair is necessary.

In the second scenario we consider a long-term timescale, on the order of the time required for nodes to traverse the whole area that they inhabit. One intuition might be that routing state has no value at this timescale, since the current topology has no relationship with the topology at the time when the route was established. However, this timescale is precisely the one considered in FRESH [3] where we have shown that one-hop routing entries, however old, can be used to constrain new route discoveries and significantly decrease the flooding overhead.

Hop-by-Hop Routing. Routing protocols typically view a route as a consistent end-to-end structure. In this model a route must be set up and converged from source to destination before data can flow across it. Clearly this is well-suited (and has been proven) to wired networks, where topology changes are rare events. For more dynamic networks, where change is the norm rather than exception, the brittle nature of this model can degrade performance. For example, a single link break can bring down an entire route, even when most of the route remains valid. As networks grow larger and routes get longer, the probability of a link break along a route increases, and the amount of time when a route is available correspondingly goes down. This reduces the overall throughput available to an application.

GREP does away with the notion of end-to-end routes and views routes as distributed structures which continuously adapt to change rather than be entirely torn down and rebuild from scratch at each topology change. In this hop-by-hop routing approach a source does not have to stop sending when a link changes along the route to the destination; in fact it is in most cases not even aware that a local repair happened further along the route.

We should also note that the exclusive use of local repairs has one drawback, namely this may result in suboptimal routes that are longer than the shortest possible path. Though our simulation results show that this is effect is not severe enough to damage GREP's performance, we believe that a worthwhile extension to GREP will allow a node to progressively 'shorten' a suboptimal route as a session goes on.

5 Conclusions

In this paper we have introduced a new approach to routing in mobile ad hoc networks, which we call *space-time routing* (STR). This approach uses both spatial and temporal distance information to determine which routing entries can be used to advance a packet towards its destination; it can be contrasted with existing protocols, which are grounded in the classic Dijkstra or Bellman-Ford algorithms and use only spatial information. We have given a general formulation and discussed possible instances of STR, including FRESH [3] and GREP [4], and discussed STR properties and loop-freedom.

We believe that STR offers many opportunities for future research. One direction of future investigation consists in exploring other STR instances, such as those outlined in Sect. 3.3. Another concerns the design of schemes to allow progressive optimization of routes computed by STR, since STR may provide routes of suboptimal length. Finally, we will consider STR in context where topology change occurs as a result of dynamics other than node mobility. For example, nodes in a sensor network usually have static positions, but topology may be dynamic as a result of duty cycling.

References

1. Perkins, C.E., Belding-Royer, E.M., Das, S.R.: Ad hoc on demand distance vector (aodv) routing. IETF, Internet-Draft (2003)
2. Johnson, D.B., Maltz, D.A., Hu, Y.C., Jetcheva, J.G.: The dynamic source routing protocol for mobile ad hoc networks (dsr). IETF, Internet-Draft (2003)
3. Dubois-Ferriere, H., Grossglauser, M., Vetterli, M.: Age matters: Efficient route discovery in mobile ad hoc networks using last encounter ages. In: Proceedings of The Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Annapolis, MD (2003)
4. Dubois-Ferriere, H., Grossglauser, M., Vetterli, M.: Generalized route establishment protocol (grep): Proof of loop-free operation. In: EPFL Technical Report IC/2003/40. (2003)
5. Boppana, R.V., Konduru, S.: An adaptive distance vector routing algorithm for mobile, ad hoc networks. In: Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and communications Societies. (2001) 1753–1762
6. Lamport, L.: Time, clocks and the ordering of events in a distributed system. In: Communications of the ACM. (1978)