Performance evaluation of routing protocols for MANETs with known connectivity patterns using evolving graphs

Afonso Ferreira, Alfredo Goldman, Julian Monteiro

Performance Evaluation of Routing Protocols for MANETs with Known Connectivity Patterns Using Evolving Graphs

A. Ferreira · A. Goldman · J. Monteiro

Abstract The assessment of routing protocols for mobile wireless networks is a difficult task, because of the networks' dynamic behavior and the absence of benchmarks. However, some of these networks, such as intermittent wireless sensors networks, periodic or cyclic networks, and some delay tolerant networks (DTNs), have more predictable dynamics, as the temporal variations in the network topology can be considered as deterministic, which may make them easier to study. Recently, a graph theoretic model – the *evolving graphs* – was proposed to help capture the dynamic behavior of such networks, in view of the construction of least cost routing and other algorithms. The algorithms and insights obtained through this model are theoretically very efficient and intriguing. However, there is no study about the use of such theoretical results into practical situations. Therefore, the objective of our work is to analyze the applicability of the evolving graph theory in the construction of efficient routing protocols in realistic scenarios. In this paper, we use the *NS2* network simulator to first implement an evolving graph based routing protocol, and then to use it as a benchmark when comparing the four major ad-hoc routing protocols (AODV, DSR, OLSR and DSDV). Interestingly, our experiments show that evolving graphs have the potential to be an effective and powerful tool in the development and analysis of algorithms for dynamic networks, with predictable dynamics at least. In order to make this model widely applicable, however, some practical issues still have to be addressed and incorporated into the model, like adaptive algorithms. We also discuss such issues in this paper, as a result of our experience.

Keywords Ad hoc wireless networks · sensor networks · evolving graphs · routing protocols · delay tolerant networks - DTNs · performance analysis

1 Introduction and Motivation

Wireless communication networks have become increasingly popular in the computing industry and are widely available in our every day life. A MANET (Mobile Ad hoc NETwork) is a collection of mobile devices that are dynamically connected in an arbitrary manner, without the aid of any established infrastructure or centralized administration [10,34]. These mobile devices with wireless transmitters are called *nodes*. When two nodes want to communicate, they may not be within each other's range, but they may communicate if other nodes between them also participate in the network, acting as routers, forwarding packets to the other end. These are called multi-hop wireless ad hoc networks.

In several environments these nodes are free to move and they may have nonuniform characteristics, driving the emergence of complex ad hoc networks that may have a highly dynamic behavior. Thus, a large number of routing protocols have been developed for MANETs [1, 33]. Besides the mobility, such protocols must deal with the typical limitations of these net-

This paper extends the initial results presented at the IEEE WiMob 2006, Montreal (Canada).

A. Ferreira · J.Monteiro
MASCOTTE Project, CNRS/I3S/INRIA, 2004 route des lucioles, B.P. 93, F-06902 Sophia Antipolis Cedex, France. *A.Ferreira is currently on leave as Head of Science Operations at the COST Office, Brussels - http://www.cost.esf.org.*
E-mail: afonso.ferreira@sophia.inria.fr
E-mail: julian.monteiro@sophia.inria.fr

A. Goldman
Department of Computer Science - University of São Paulo - Rua do Matão 1010, São Paulo, SP, 05508-090, Brazil
E-mail: gold@ime.usp.br

works, like energy limitations, low processing capacity, low bandwidth, and high error rates [10].

There are different approaches which try to optimize the cost of a routing path, but, until recently, most of them did not take into account the fact that some MANETs may have known connectivity patterns, as in DTNs [17, 38] such as LEO satellite networks [8, 14] and Wireless Sensor Networks (WSNs), where, due to energy limitations, network nodes can be scheduled to sleep in given periods [1, 19, 31, 35].

In this kind of networks, the topology dynamics at different time intervals can be predicted (see Fig. 1). Therefore, the performance evaluation of routing protocols should be easier, although a formal tool for benchmarking such protocols has yet to become a standard.

1.1 Our contribution

In this paper we use *evolving graphs* (EG) – a formal abstraction for dynamic networks [6, 13] –, in order to design and evaluate least cost routing protocols for MANETs with known connectivity patterns. These protocols are then used as benchmarks for establishing fair comparisons between the four MANET routing protocols, namely DSDV [24], DSR [18], AODV [25] and OLSR [16]. This is done through extensive simulations using $NS2$ within different scenarios. It is important to note that the previous protocols were designed to work within connected networks, as they are based in store-and-forward techniques, and not on store-carry-forward as the EG techniques. However, the EG routing protocols can be used as a lower bound reference.

We note that the algorithms and insights previously obtained through the EG model are theoretically very efficient and intriguing. The central objective of our work is thus to assess the usages of these theoretical results in practical situations, where packet dropping, for instance, may pose unexpected challenges to the EG algorithms. As an example, although we do not limit the buffer size on the nodes, we do propose an extensive discussion on the nodes' transmission queues.

The remainder of this paper is organized as follows: After presenting the related works, in the next section we describe the concept of *evolving graphs*. The routing protocols to be compared are defined in Section 4. Section 5 shows the simulation environment and the decisions made in the implementation. Section 6 and 7 present the simulation results and analyses. We close the paper with our conclusions and avenues for future research in Section 8.

2 Related Work

It is interesting to note that the theory of Evolving Graphs is contemporary with the formulation of DTNs, although they followed different objectives. The EG theory formalised in a graph theoretical framework the concept of networks with known connectivity patterns. It focussed both on graph theory structural properties and on routing issues related to the optimisation of packet delivery between two nodes, using one of the following criteria: foremost, shortest, or fastest journey [2, 6, 13].

On the other hand, research done in the field of DTNs usually assume only partial, or even no knowledge about future connectivity patterns [11,38]. Noticeably, however, some research done in DTNs do assume known connectivity patterns, as in [17], where the authors propose a Linear Programming solution when full future connectivity knowledge is available. Some other algorithms which require less knowledge were also presented. For the sake of conciseness, we will refer henceforth to networks with full future connectivity knowledge as deterministic DTNs, or DDTNs for short.

Another difference is on the model, where bounded buffers were considered on the nodes. In our work, we assume unbounded buffers, although we use bounded buffers for message transmission. In [29] the authors also propose a deterministic solution based on a tree construction, which considers in a simplified way the message transmission time.

In [32] the authors studied in detail the single-copy case, that is, for a message transmission there is only one copy of the message on the network, which is the same assumption used in the EG model, as described in this paper. They present an "Oracle-based" optimal algorithm and several other partial knowledge algorithms. In their paper there were no details on the possible congestion of the "Oracle-based" algorithms, which is one of the main contributions of our work presented here.

Another work which considers predictable behavior is [21] that presents a shortest delay path routing with a complete knowledge of future connections. It also compares this algorithm with Hot potato, Most Frequent Neighbor and Epidemic Routings. However, there is only a very small experiment on the delays with bursty traffic.

Finally, a close approach aiming at benchmarking routing protocols in dynamic networks is the MERIT framework [12]. It uses the notion of competitive analysis [9] on a dynamic setting in order to assess the quality of protocols studied on snapshots describing the history of the network. The results proven include finding

a sequence of paths that connect a given pair of nodes throughout the system, such that the global routing plus re-routing costs are minimized.

3 Evolving Graph Model

The *evolving graph* (EG) model, proposed in [13] aims to represent a formal abstraction of dynamic networks, through the formalisation of a time domain in graphs.

As an example, consider the four snapshots taken at different time intervals of a MANET, as depicted in Fig. 1. As one can readily observe, nodes D and G are never connected on a single time interval. Notwithstanding, D can indeed send messages to G, using the *path over time* composed of D, C, E, F, G. Surprisingly, this otherwise trivial fact cannot be directly modeled by usual graphs.

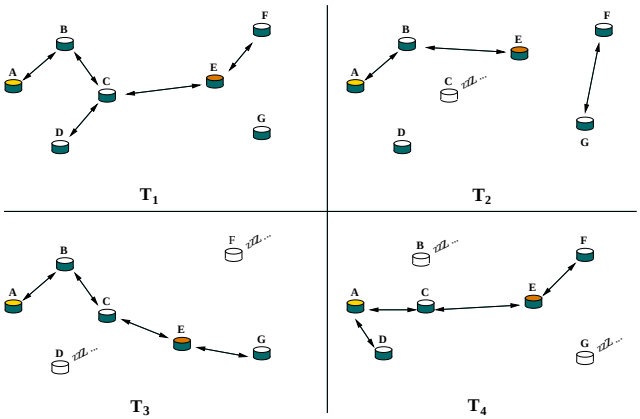


Fig. 1 The evolution of a MANET over time. The indices correspond to successive snapshots in time. "Zzz" indicates a sleeping node.

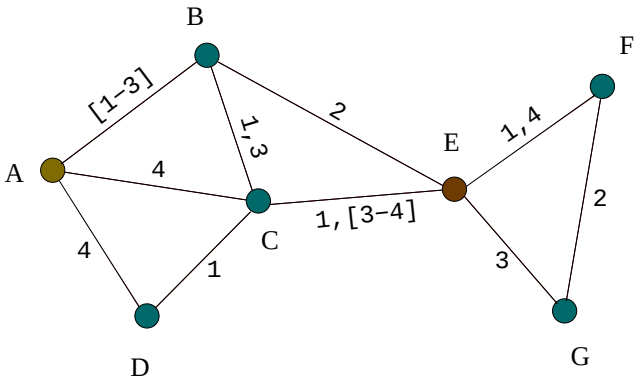


Fig. 2 Evolving graph corresponding to the MANET in Fig. 1. Edges are labeled with corresponding presence time intervals. Observe that $\{E, G, F\}$ is not a valid journey, since the edge $\{G, F\}$ exists only in the past with respect to $\{E, G\}$.

Concisely, an evolving graph is an indexed sequence of τ subgraphs of a given graph, where the subgraph at a given index corresponds to the network connectivity at the time interval indicated by the index number, as shown in Fig. 2.

The time domain is further incorporated into the model by restricting *journeys* (i.e., the equivalent of *paths over time*) to never move into edges which only existed in past subgraphs. A journey in an evolving graph is thus a path in the underlying graph whose edge time-labels are in a non-decreasing order. Now, it is easy to see in Fig. 2 that D, C, E, F, G is a journey, as mentioned above. Further, note that D, C, E, G is also a journey, with less hops, but delivering the message later (in time interval 3 instead of 2), giving raise to different objective functions that may be optimized.

3.1 Journey Metrics

In the pursue of an optimal journey in networks with known connectivity patterns, three metrics have been formalized until now for EG [6]. They are the *foremost*, *shortest*, and *fastest journey*, which find, respectively, the earliest arrival date, the minimum number of hops, and the minimum delay (time span) route. These three parameters can be individually optimized in polynomial time [6].

We use in this paper the *Foremost Journey* algorithm, which computes from a source node s the journeys that arrive the earliest as possible on all other nodes. The algorithm to compute such journeys can be seen as an adaptation of *Dijkstra*'s shortest paths algorithm [9], and is detailed below.

3.2 Foremost Journey Algorithm

Remind that, in order to compute shortest paths, the usual Dijkstra's algorithm proceeds by building a set C of *closed* vertices, for which the shortest paths have already been computed, then choosing a vertex u not in C whose shortest path estimate, $d(u)$, is minimum, and adding u to C, i.e., closing u. At this point, all arcs from u to $V - C$ are *opened*, i.e., they are examined and the respective shortest path estimate, d, is updated for all end-points. In order to have quick access to the best shortest path estimate, the algorithm keeps a min-heap priority queue Q with all vertices in $V - C$, with key d. Note that d is initialized with ∞ for all vertices but for s, which has $d = 0$ (in terms of routing protocols, d needs to be initialized with the current time t).

The main observation in Dijkstra's method is that prefix paths of shortest paths are shortest paths them-

selves. Unfortunately, the prefix journey of a foremost journey is not necessarily a foremost journey (e.g., considering the EG in Fig. 2, a message sent at time interval 1 from A to G can use the journey A, B, E, G, with the packet reaching G at time interval 3. The prefix journey A, B, E will reach E at time interval 2, although this is not a foremost journey from A to E, which is in fact A, B, C, E, arriving at moment 1). On the other hand, it was proven that there exists at least one foremost journey with such a property in an evolving graph [6, 13].

To compute the *Foremost journey* starting at time t from s to all other nodes, we use a direct adaptation of *Dijkstra*, sketched below, as detailed in [6]:

1. Set $d(s) = t$, and $d(u) = \infty$ for all other nodes.
2. Initialize *min-heap* Q, sorted by d, with only s in the root.
3. While Q $\neq \emptyset$ do
   (a) $x \leftarrow$ root of heap Q.
   (b) Delete the root of heap Q.
   (c) For each open neighbor v of x do
      i. Compute first valid edge schedule time greater or equal to current time step
      ii. Insert v in the heap Q if it was not there already.
      iii. If needed, update $d(v)$ and its key.
   (d) Update the heap Q.
   (e) Close x. Insert it in the foremost journeys tree.

At the end, we have a tree with the *Foremost journey* traversal paths from s (starting at time t) to all other nodes.

Note that the computation of the first valid edge schedule done at the inner loop may take into account the traversal time for the edge, i.e., the duration of the transmission, if needed. This is the case of *timed* evolving graph [13].

A foremost journey from a source node s to all other nodes can thus be computed in $O(M(log\delta_E + logN))$ time, where N is the number of vertices, M is the number of edges and δ_E is the maximum number of presence time intervals over all edges. The term $log\delta_E$ stems from the lookups into the schedule list of intervals, which is required to decide the earliest time interval in which to cross each visited edge.

The routing protocol originated by this algorithm will be henceforth referred to as $EG_{Foremost}$ and is detailed in next section.

4 Routing Protocols for MANETs

A great deal of work has been produced comparing the performance of the four main MANET routing protocols, namely DSDV, DSR, AODV and OLSR, that were designed to provide routes in connected networks [4, 5, 10, 16, 26].

The first of these routing protocols, Destination-Sequenced Distance Vector (DSDV) described in [24], is a *proactive* table-driven protocol based on the distributed Bellman-Ford algorithm, with loop-freedom improvement. Each node has a routing table for all reachable nodes, which stores for each destination the next-hop, the number of hops, and a sequence number. DSDV requires periodical flooding to update the routing table.

Dynamic Source Routing (DSR) [18] is a *reactive* protocol, allowing nodes to dynamically discover a route to destination, on demand. Such routes are stored in a route cache to enhance the performance. Source routing means that each packet carries in its header the complete ordered list of nodes (the path) to the destination, so that the forwarding nodes do not need to have the routing information. There is a clear compromise between the size of routing tables and packet size.

The Ad-hoc On-Demand Distance Vector Routing (AODV) [25] is based on DSDV and DSR. AODV is also a *reactive* protocol, which requests a route when needed, and maintain a traditional routing table to destinations in use. A routing table entry is *expired* if not used recently.

Finally, Optimized Link State Routing (OLSR) [16], is a *proactive* table-driven protocol and inherits the use of link state algorithm, using shortest path first forwarding. It periodically exchanges the topology information with neighbors, and every node maintains the topology of the whole network. To minimize flooding, OLSR uses nodes that act as Multi Point Relays (MPR). Only these special nodes are responsible for forwarding control traffic. As DSDV, this is a proactive protocol, so the routing paths are available immediately when needed.

There are many other routing protocols [1, 34] with specialized characteristics. We are not going to evaluate them here, mainly because this experiment aims to compare $EG_{Foremost}$ with massively tested and analyzed protocols, as are the four above.

4.1 The $EG_{Foremost}$ protocol

One of the objectives of this work is to investigate the behavior of the EG foremost algorithm as a theoretical optimal routing protocol. In this respect, it is important to mention that its implementation as a distributed routing protocol, i.e., with control messages to distribute the EG, keeping it up to date and fail safe mechanisms, is out of the scope of this paper. Therefore, we assume that all nodes have the knowledge of

Table 1 Edge schedules for the EG in Fig. 2.

Node pair	Edge schedules
A – B	1, 2, 3
A – C	4
A – D	4
B – C	1, 3
B – E	2
C – D	1
C – E	1, 3, 4
E – F	1, 4
E – G	3
F – G	2

the *EG*, which makes straightforward the implementation of the protocol. This is still true if the assumption holds for transmitting nodes only.

Let *edge schedules* be a set of time intervals representing the existence of link-connectivity between two nodes. An *edge* exists when two nodes are in the range of each other. The evolving graph of a dynamic network can be represented by a list of *edge schedules* for each pair of nodes. Thus, each node has a list of its neighbors at a given time (as detailed in Section 5.2, the *EG* is calculated from the mobility scenario and a well known transmission range).

When a packet arrives at the routing layer of node u at time t_{now}, the node computes the *foremost journey* (as shown in Section 3.2) from the packet source to its destination.

Suppose that the journey next hop is the node v at time t_v. If $t_v = t_{now}$, then both nodes are in the range of each other (i.e, there is an edge in *edge schedules* of (u, v) at time t_{now}), and the packet is readily forwarded to v. Otherwise, if $t_v > t_{now}$, the nodes are not reachable, and the node u must schedule the transmission of the packet to the time t_v ahead. This is the earliest feasible edge present in *edge schedules* of (u, v) with time greater than t_{now}.

In Table 1 we show the corresponding *edge schedules* as shown in the example of Fig. 2. Note that the *edge schedules* are the presence time intervals at the labels.

As an example, the foremost journey for a message sent at time index 1 from D to G will be D, C, E, F, G. The packet will reach F at the same time index 1, then afterwards F will schedule to send the packet to G at the earliest edge presence in the *edge schedules* of $F-G$. Hence the packet will be sent by F at time index 2.

If two routes have the same time length when computing the *foremost journey*, the one with less number of hops will be chosen for routing, and if they even have the same number of hops, the route with the smaller node ID will be chosen. This ensures a total order when choosing nodes (i.e. sorting nodes in the heap).

In the examples above, the edges traversal times are not taken into account, for the sake of a simpler illustration. In the implementation and in all simulations the edge traversal time was indeed estimated (we include a discussion about it in Section 7.1)

5 Simulation Environment

We have conducted our performance analyses using the *NS2* [23] simulator version 2.31, with the mobile extensions by CMU Monarch which provides IEEE 802.11 Medium Access Control (MAC) protocol [15], and realistic radio and physical layer with the Two Ray Ground propagation model. The radio model uses characteristics similar to the 802.11g standards, modeled as a shared-media radio with a nominal bit rate of 54 MB/s and an omni-directional antenna with nominal propagation range of 50m. The RTS/CTS radio scheme was turned on in our experiments.

In the simulations, *60 nodes* are randomly placed in terrains with size varying from 300m x 300m, 300m x 200m and a larger one with 1000m x 1000m area, these parameters lead to different density of nodes and are discussed ahead. The simulation time is *3000 seconds*, and the first 1000 seconds of each simulation are not considered, for the sake of the stability of the movements [37]. A number of *10 constant bit rate (CBR) UDP traffic flows* are chosen between node pairs (nodes 1 to 10 are the transmitters and nodes 40 to 49 are the receivers). The average *traffic rate is one packet/sec*, with 256 bytes long packets. Each flow starts to generate packets at random at instant 1000 seconds of simulation and remains transmitting until the end. This low data rate is chosen to address a sensor network-like environment, where dedicated sensor nodes are constantly collecting data. The interface queue length at link layer (IFQ) was doubled from the default 50 packets to 100 packets at each node. We do not use TCP for the simulations, as we did not want to investigate TCP particularities, which uses flow control, retransmit features and so on. We are solely interested in the behaviour of the routing protocols.

In these experiments (in contrast with [22]) we decided to disable the ARP (Address Resolution Protocol) of mobile nodes, agreeing with the arguments in [7] by Carter, Yi and Kravets, in that MANETs need to have their own ways of neighbour discovering. This is because, in *NS2*, the address resolution is an approximation of the BSD Unix ARP, so the resolution is performed on-demand as packets arrive from the application layer, and the buffer size to each neighbor is only one single packet. When an address resolution is

in process, all incoming packets to the same destination will be dropped. This leads to a great amount of dropped packets in ARP mode.

Therefore, in our simulations we assume that each node has the IP and MAC addresses of its neighbours.

5.1 Mobility Models

We divided our experiments into two separate kinds of scenarios. One uses the popular *Random Waypoint* (RWP) model [18], while the other uses a new mobility model, called *Intermittent Model*, which is more suitable to the case of WSNs and is explained below.

5.1.1 Random Waypoint Mobility Model

In the *Random Waypoint* (RWP) mobility model, a mobile node moves to a randomly chosen location, with speed randomly chosen from 1 to 3 m/s according to a uniform distribution, and pauses for a uniformly chosen time between 0 and PauseTime. The simulation was run with values of PauseTime varying from 0 (continuous motion) to 1500 seconds (very low mobility in the network). To avoid known problems of the RWP model, as shown in [37], we are using only non-zero values of minimum speed. The use of this classical scenario, yet with its known limitations, is important to compare the results with other performance studies. The program *BonnMontion* from the University of Bonn [3], was used to generate these scenarios.

5.1.2 Intermittent Mobility Model

This mobility model is based on fixed nodes whose positions are chosen randomly with uniform distribution. The nodes remain uninterruptedly turning themselves *on* and *off* (awake and sleep) in given periods (see Fig. 3). Here, the parameters we change are the SleepProb (ranging from 0 to 50%), and the HoldTime (ranging from 15s to 180s). In the beginning of the simulation each node is awake, and for the entire simulation it has a SleepProb probability to go to sleep (turning itself off). If a node goes to sleep, it remains off for a uniformly randomly chosen HoldTime. Once this time expires, the node is turned on and stays awake for another period based on HoldTime. If a node does not go to sleep (probability of $1 - \text{SleepProb}$), it will stay awake for HoldTime before trying again. It is important to point out that HoldTime is recomputed at each state change.

This new model aims to capture the behavior of networks with functioning schedules, like wireless sensor networks. Chapter 7 of [19] presents several related node models.

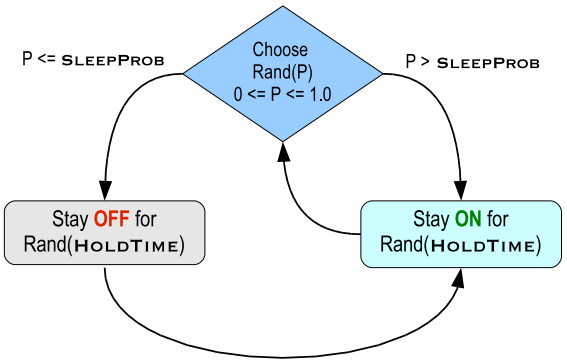


Fig. 3 Lifecycle of a node in the Intermittent Mobility Model.

With the aim of pursuing a constant traffic rate in the network, the nodes which generate the traffic flows never go to sleep during the whole simulation.

In both models, for each evaluated parameter we created 20 random scenarios with different random seeds. Therefore, we ran the simulation 320 times for the first model, i.e., 20 times for each value of PauseTime: $0, 100, 200, 300, ..., 1500$ seconds, and 440 times for the *Intermittent Model*, i.e., 20 times for each combination of SleepProb: $0, 5, 10, ..., 50\%$ and HoldTime: $15, 180$ seconds. Note that in the second scenario we change two parameters, and the value of 0% of SleepProb means that no one goes to sleep, hence the network remains static from the beginning to the end of the simulation, which is relevant as a reference. All five routing protocols (AODV, DSDV, DSR, OLSR, $EG_{Foremost}$) were run on the same 760 scenarios. Thus, identical mobility and traffic scenarios (as they have the same seeds) were used across protocols.

5.2 NS2 Implementation Details

Each simulation in *NS2* generates a trace file, containing all communications that have been done between nodes, including the MAC layer. These files were analyzed to consolidate the results, which are shown in Section 6.

The implementations used to evaluate DSDV, AODV and DSR protocols are the ones provided in the *NS2* package. All of them were implemented by the CMU Monarch group [27]. Concerning OLSR, we used the implementation by Francisco J. Ros [28] (UM-OLSR) version 0.8.8. In all of them we used the default parameters and constants. The AODV implementation, as of *NS2* version 2.31, contains some further optimization codes from [26].

In *NS2*, the node states "sleeping" and "awaken" used in the *intermittent model* scenarios are implemented using the commands *on* and *off* already implemented in the mobile agent. But the last version of *NS2* did not work correctly with this command. To ensure correction, we removed the line that do a `reset-state` in the command `off` in the `mobilenode.cc` file, which is only used by direct diffusion agent.

In order to generate the *evolving graph* that will be used as input to our protocol, we use the following strategy. Mobility in *NS2* is usually represented by a script in Tcl language containing scheduling commands, e.g. *setdest*, *on*, or *off*. In general it is saved in a separate file used by the simulation. Therefore, we wrote a program (*calceg*) that reads a mobility file used by *NS2* and captures the node movements to generate a corresponding *EG*. This *EG* is afterwards used as input for the foremost journey *EG* based protocol ($EG_{Foremost}$), as shown in section 3.2. Note that, to calculate the *EG*, the transmission and reception range of each node must be taken into account; in our case is fixed at exactly 50 meters . We build our EGs from the mobility models after a post mortem analysis in a continuous way. It does not use any discretization technique. We calculate analytically the exact moments where the nodes enter in the communication range of each other, and also when they are not able to communicate anymore.

Before the simulation begins, this *EG* of the network is distributed among all nodes. Each node in the simulated network knows the connectivity pattern of the network during the simulation. This is important for benchmarking purposes, since the EG may be generated and used as a reference when developing or tuning routing protocols and mobility models. Furthermore, there are many practical situations, like those shown in [1, 8, 14, 20, 36], in which an *EG* can be built before the routing phase.

From a theoretical perspective, the EG-based protocol can be considered as a distributed protocol, since there is no central controller. The fact that every node possesses the full description of the EG is part of the set of our hypothesis. If each node has local knowledge about future connectivity, then global knowledge might be obtained by a dissemination mechanism like the one deployed by OLSR. If this is the case, then a careful study of the impact of this routing overhead should be conducted.

[0] The EG implementation and related software can be found at http://www.ime.usp.br/ jm/mobidyn/software.

6 Simulation Results

In this section we show the results obtained by simulation of a DDTN composed of wireless mobile nodes that move around, go to sleep for a while, and communicate with each other.

As in the case of the mobility models, the results shown here are separated in two parts, one using the RWP mobility model, and another using the *Intermittent Model*.

We focused our analysis in four main metrics:

- **Average throughput**: The average number of packets received per amount of time (from the first packet sent to the last packet received);
- **Average end-to-end delay**: The average time between sending and successfully receiving a packet;
- **Ratio of dropped packets by no route (NRTE)**: Fraction of dropped packets by *no available route* per total number of sent packets;
- **Ratio of dropped packets by Interface Queue overflow (IFQ)**: Fraction of dropped packets by *link queue overflow* per total number of sent packets. This queue is at the link layer, i.e., it is used when the routing layer wants to effectively send a packet to be delivered.

Error bars on the figures indicate a 95 percent confidence interval.

6.1 Random Waypoint Mobility Model

As mentioned earlier, in the RWP scenario the control parameter is the PauseTime. Low values of PauseTime mean *high* mobility and high values of PauseTime mean *low* mobility.

As shown in Fig. 4, the $EG_{Foremost}$ performance has the lowest values of *dropped ratio* for all pause times, followed closely by reactive protocols. With low mobility (high pause times), the number of dropped packets raise to 5.7% of the transmitted packets. DSR has good values compared to others and it is surprising that AODV does not perform well (i.e. better than DSR) at high mobility values, in contrast to what was as shown by Perkins, Royer, Das, and Marina in [26]. This behaviour may be explained by the very low network load of our simulation (1 packet/s with 10 traffic sources).

The pro-active, table-driven protocols do not perform well in this first scenario, since the periodically updated tables do not get updated as fast as needed. The drop-ratio for DSDV and OLSR are the worst, ranging from 60% in high mobility to 39% in low mobility simulations, showing that they are very sensitive to mobility. They perform better when the mobility decreases.

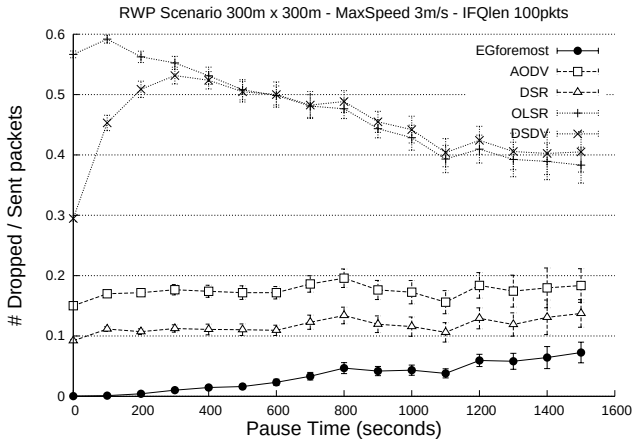


Fig. 4 Drop ratio as a function of PauseTime (mobility).

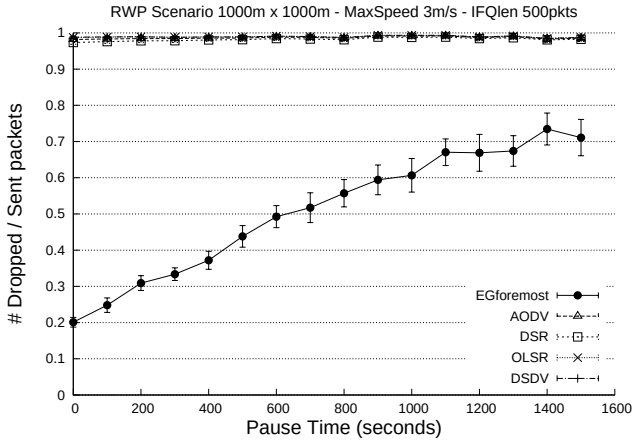


Fig. 5 Drop ratio as a function of mobility using a low density of nodes scenario.

Finally, $EG_{Foremost}$ produced the best results in this metric in all RWP simulations scenarios. The theoretical throughput of the network is $1 * 256 * 8 = 2048$ bits/s and the $EG_{Foremost}$ results are very close to it.

As stated before, the $EG_{Foremost}$, as expected, performed well in the above scenarios, comforting our claim that it may serve as a benchmark when evaluating the performance of other protocols on similar mobility models.

Scenarios with low density of nodes

In the above scenarios, a terrain with 300m x 300m area was used. The coverage ratio, i.e. the ratio between the sum of nodes transmission ranges divided by the field area was 5.2. To simulate a scattered scenario we raised the terrain size to 1000m x 1000m, which leads to a coverage ratio of 0.47. Using this scenario we can measure how routing protocols behave when nodes remain disconnected for a long time, which is generaly the case of DTNs.

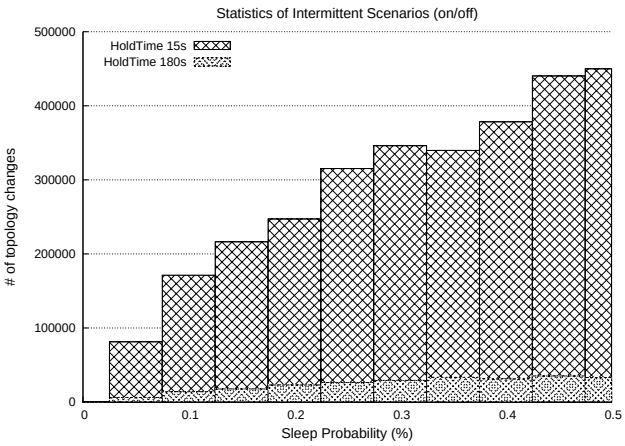


Fig. 6 Number of changes in the network topology in the Intermittent Model scenario for different values of SleepProb.

In Fig. 5 one can see that regular ad hoc protocols are not adapted to a disconnected environment. However, the $EG_{Foremost}$ protocol shows that a high amount of packets could still be delivered. In the high mobility scenarios the drop ratio was close to 20% and it increases to 70% of drops in the low mobility scenarios. It should be noted that the delivered packets have a high average end-to-end delay, ranging from 285 seconds with PauseTime 0 to 606 seconds with PauseTime 1600s.

6.2 Intermittent Model

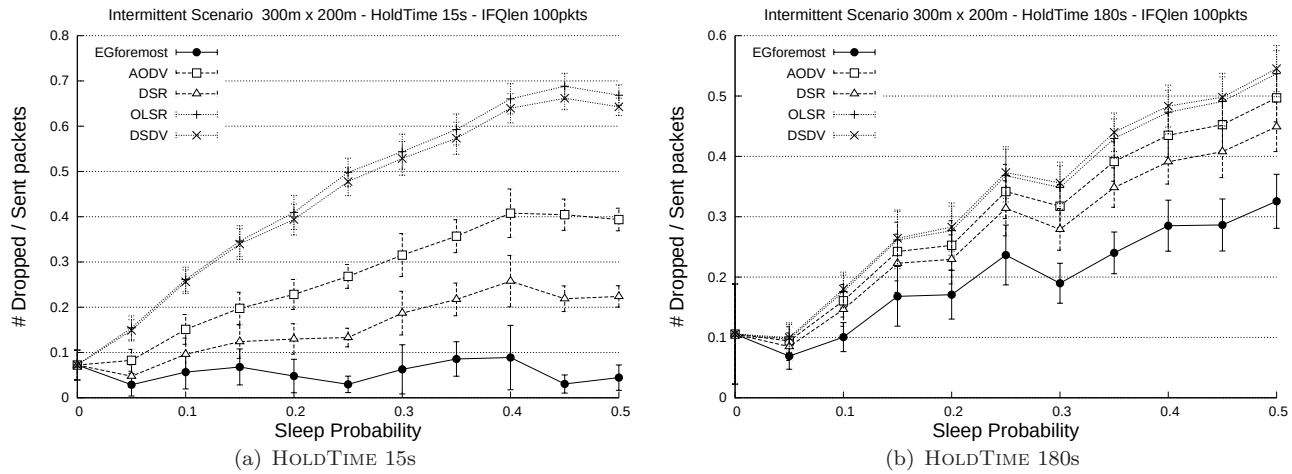
The intermittent scenario is well adapted to DDTNs, since the nodes' on/off dynamics are easily predicted or even pre-programmed.

We changed the values of SleepProb from 0 to 50%. High probability to sleep means that the network has a low connectivity, i.e., a large quantity of nodes are disconnected from each other because many of them are in sleep state.

The values of HoldTime control how slow the nodes change their states (on/off) or, in other words, *how often connections among nodes change*. Low values of HoldTime means high dynamics and vice versa. Fig. 6 illustrates this behavior of the Intermittent Model scenario. In the low connectivity scenarios (SleepProb at 50%) the number of topology changes for a HoldTime of 15s is 12 times the number of those with HoldTime 180s.

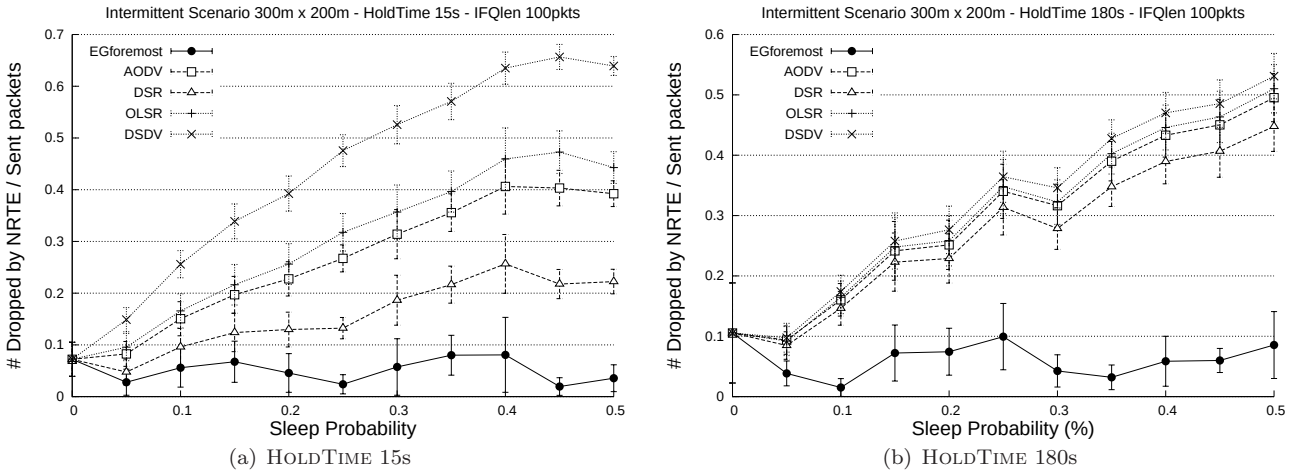
As we will see in the following, this model harness some characteristics of connectivity dynamics that are not captured with the former RWP model.

After some experimentation of the simulation parameters, we reduced the size of the simulation field to a retangular 300x200m, so the coverage ratio is raised



(a) HoldTime 15s (b) HoldTime 180s

Fig. 7 Total drop rate as a function of SleepProb (connectivity).



(a) HoldTime 15s (b) HoldTime 180s

Fig. 8 Drop packets by NRTE ratio as a function of SleepProb (connectivity).

to 7.85. Otherwise, the number of dropped packets was very high, as the nodes in these intermittent scenarios do not move.

In Fig. 7, we show again that the $EG_{Foremost}$ has better values of *drop rate*. Observing the results, in the case of high dynamics (HoldTime equal 15s), the drop rate of $EG_{Foremost}$ is close to zero in all connectivity scenarios. On the other hand, in the case of low dynamics (Fig. 7(b)), the values of drop rate for EG decrease, with the other protocols, as the connectivity decreases (from 0 to 50% SleepProb). This $EG_{Foremost}$ loss of performance (reaching 32% less if compared to the rate of flow) is not related to non-existing routes, because, as shown in the graphics of Fig. 8, the number of packets dropped by NRTE is very low (an average of 5% on the both scenarios).

The values of $EG_{Foremost}$ in Figs. 8(a) and 8(b) show that the number of *dropped packets by NRTE* is a lower bound in this metric, i.e., when $EG_{Foremost}$ drops a packet by NRTE, it means that the requested path does not exist in any moment of time. Therefore, $EG_{Foremost}$ may again be used as a benchmark to measure how good the other protocols are performing.

The increase of the total drop rate on low connectivity scenarios is not due to inexistent routes, but to other reasons analysed below.

7 Further analyses and improvements

The goal of the foremost journey algorithm is to calculate journeys that reach the destination as soon as possible. However, in this process some packets may wait for a long time for a connection to be established, and this waiting time is computed in the end-to-end delay metric. In contrast, in the simulation of the other

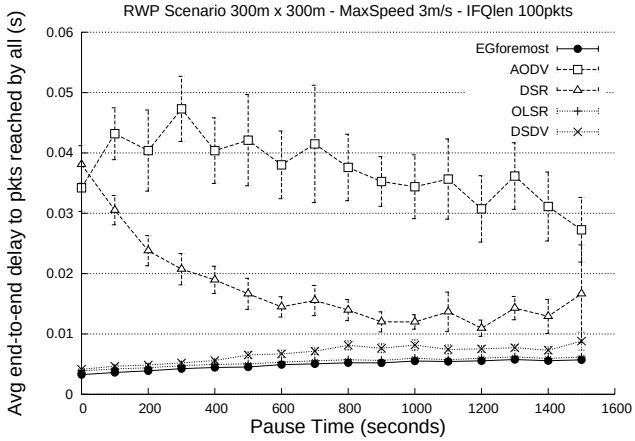


Fig. 9 Average end-to-end delay of packets successfully delivered in all protocols.

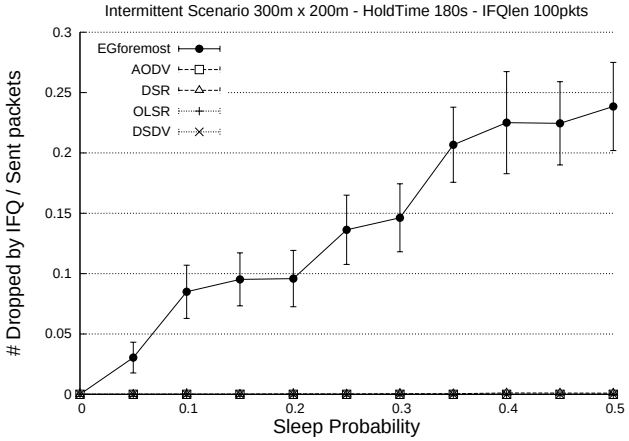


Fig. 10 Number of dropped packets by IFQ overflow on a HoldTime 180s scenario.

protocols some of these "late" delivered packets are just being dropped and do not contribute to the end-to-end delay count. In other words, when using the Foremost Journey EG based routing algorithm, the packets end-to-end average delay is usually larger, even though it was proven in [6] that $EG_{Foremost}$ ensures that the packets will reach the destination as soon as possible if a journey exists in the network.

This gives the opportunity to add a new parameter *MaxDelay* in the EG algorithms, which is the maximum delay time that a packet could wait to be delivered in the DDTN. If the calculated delay time is greater than *MaxDelay*, the packet could be dropped instead of overflowing the network.

Now, remind that almost all packets are delivered by the $EG_{Foremost}$ protocol (see Fig. 4). Hence, to be fair with the foremost algorithm and better perceive the performance of the $EG_{Foremost}$ protocol, we calculated the average end-to-end delay taking into account *only the packets that have been successfully delivered* at the destination in all protocols (i.e. the intersection of received packets). The results in the Fig. 9 shows $EG_{Foremost}$ as a lower bound in the *end-to-end delay* in this scenario. The reactive protocols, AODV and DSR, do not have a smooth curve, due to the induced delays from the route discovering process. The variance of their values is very high too. It is important to point out the good performance of the OLSR protocol, very close to that of the $EG_{Foremost}$.

7.1 Bottlenecks and Congestion

The intrinsic behavior of $EG_{Foremost}$, namely to schedule packets to be sent when some connections are established, yields the problem of *bottlenecks* [30], since a large quantity of packets are scheduled to be sent at the same moment, and the link interface queue (IFQ) cannot hold that incoming traffic (its size is 100 packets). Furthermore, the *EG* algorithm do not have any mechanism to balance the flows and many flows with different sources could potentially use the same path, even when some other ones are available at the same cost (i.e. arriving at the same time).

We note that the simulation done using the *random waypoint model* does not suffer from this effect. Due to the high mobility of such a scenario, the *bottlenecks* do not show expressive values. This characteristic appeared in the low connectivity and low dynamics scenarios of the *Intermittent Model*, in which the nodes in the evolving graph remain disconnected for long time periods, and a large quantity of packets are then scheduled to the moment when these nodes wake-up.

In Fig. 10 we see the high values of *dropped packets by IFQ overflow* on a low connectivity scenario (24% of packets are dropped by IFQ at SleepProb 50%).

The histogram in Fig. 11 shows the number of *dropped packets over time* for one single simulation, namely with SleepProb at 50% and HoldTime at 180s. It shows that in $EG_{Foremost}$ the packets are dropped in a burst, again because important nodes go to sleep for a long time and when they wake-up, a large quantity of packets are waiting to be sent, overflowing the queues that then drop the packets.

7.2 Congestion with varying flows over time

For the sake of clarity and to allow for comparisons, we show below the results of some experiments using a variable offered load on the network, since the previous experiments have very low traffic demand.

The same parameters from the intermittent model scenarios were used, but the traffic flow rate was changed

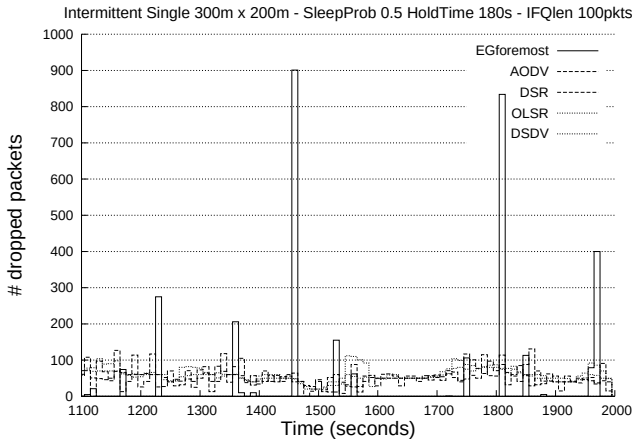


Fig. 11 Number of dropped packets over time for one single Intermittent Scenario (SleepProb. 50% and HoldTime 180s).

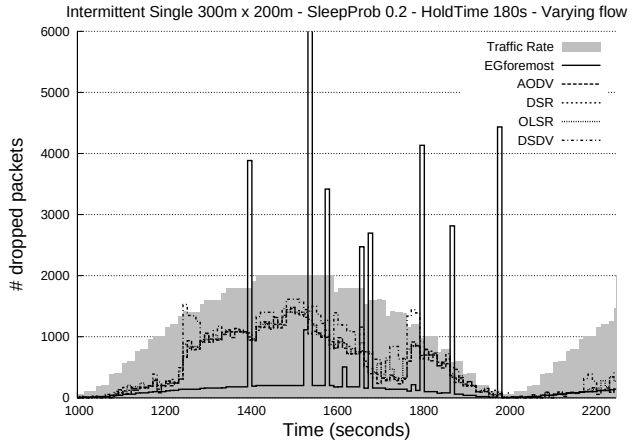


Fig. 12 Number of dropped packets over time for one single Intermittent Scenario with varying flow. The SleepProb value is 20% and HoldTime is 180 seconds.

during time according to a non negative senoidal curve, with discretized values of CBR chosen from the following rates: 0.5, 1, 2, 4, 6, 8, 10, 12, ..., 20 packets/second. The gray shaded area in Fig. 12 represents the rate of traffic generated in the network by the 10 traffic sources.

Again, the number of dropped packets by $EG_{Foremost}$ is the one with high peaks, and with the increase of traffic rate its behaviour is even worse. Note that, even at time 2000 seconds, when the traffic rate is very low (one packet every 2 seconds) the number of drops is high. This is due to many packets that are scheduled to be delivered at that same point in time. Almost all packets dropped by the EG protocol resulted from IFQ overflow.

7.3 Reducing congestion in $EG_{Foremost}$

The discussion above shows the importance of managing the flows of data during time, even when using EG protocols as a reference. Unfortunately, balancing flows in evolving graphs is still an open problem in Graph Theory. Therefore, we tried three different empirical approaches to address the packet dropping problem caused by bottlenecks in $EG_{Foremost}$ (see Table 2, below):

Table 2 Approaches used to minimize the bottlenecks (Fig. 13).

Packet	IFQlen 500	Jitter 0.5s	SmartJitter
p_1	t	t + rnd(0.5)	t
p_2	t	t + rnd(0.5)	t
...	t	t + rnd(0.5)	t
p_{50}	t	t + rnd(0.5)	t
p_{51}	t	t + rnd(0.5)	$t + \delta$
p_{52}	t	t + rnd(0.5)	$t + 2 * \delta$
p_n	t	t + rnd(0.5)	$t + (n - IFQlen) * \delta$

1. **Jitter**: Add an enforced jitter (a random value uniformly chosen between 0 to T seconds) at sending time to each packet. We experimented with values 0.1 and 0.5 of T;
2. **SmartJitter**: Add a fixed size jitter only when some connection is overflown. The size of this jitter is the same as the average edge traversal time;
3. **Increase the IFQ length**: Raise buffer size of the interface queue from 100 to 500 packets. We also showed, as a reference, the values with IFQ length of 50 packets.

The results of the experiments with these three approaches can be seen in Fig. 13. IfqLen 100pkts is the reference curve, as seen in Fig. 10. The error bars were removed from this figure to increase readability, their values were between 0.01 and 0.04, and were similar among the experiments.

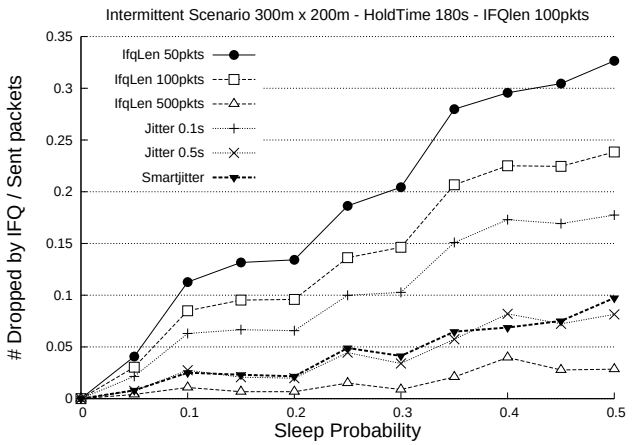


Fig. 13 Number of dropped packets by IFQ overflow with different solutions to minimize the drop rate (enforced jitter, smart jitter and raise de IFQ length to 500pkts).

The first approach is the enforced random chosen jitter when sending each packet at each node, ranging from 0 to one tenth of a second (0.1s) and half of a second (0.5s). In the average, the number of dropped packets decreased 32% and 75% respectively. One drawback of this approach is the high values of end-to-end delay, due to the extra time added at each scheduled packet.

The second approach, the SmartJitter, is an improvement of the former. Here, when the queues (one for each pair of neighbours) are not full, the nodes can send packets to other nodes without any jitter on sending time. However, when some node fills the IFQ, then a fixed size jitter is added to each subsequent packet to be sent. The calculation of the SmartJitter time at node u sending a packet to v at time t_v is shown in the following schema:

if $npkts[t_v] \leq IFQlen$ **then**
  $smartjitter = 0$
else
  $smartjitter = \delta * (npkts[t_v] - IFQlen)$
end if
$npkts[t_v] = npkts[t_v] + 1$

At the end, the sent time will be $t_v = t_v + smartjitter$ and $npkts[t_v]$ is the number of packets already scheduled to use the edge from u to v at time t_v. $IFQlen$ is the interface queue length, and δ is the average traversal time of one-hop transmission. We used the value 3.6 ms in all simulations, which was estimated from the average value of one-hop transmission with packet size of 256 bytes. The value of the traversal time is linearly dependent on the size of the packet. This value was obtained through simulations using similar network loads. The traversal time used in the EG foremost algorithm is the same estimated δ value.

With the SmartJitter, the number of dropped packets also decreased 75% compared to the original $EG_{Foremost}$. The values reached by the SmartJitter are similar to ones with the enforced jitter of 0.5s, thus, in the former case the average end-to-end delay is 10% less than the enforced jitter.

We note that the best solution is to increase the default length of the IFQ from 100 to 500 packets. In this case, the values of dropped packets decrease 92%. This shows that the SmartJitter is a good solution when the size of IFQ buffer cannot be changed.

Finally, we tested the introduction of the SmartJitter at the first scheduled packet (instead of waiting for the queue to become full). However, the end-to-end delay increased, while the number of packets dropped remained roughly the same. This is the reason why we do not report these results here. As a matter of fact, if the number of *npkts* is very high compared to the queue size, a packet could be scheduled to be sent a long time after its original time t_v, at which point the edge could not be there anymore.

8 Conclusion

Our contribution in this paper is to show that an *EG* based routing protocol is well suited for networks with known connectivity patterns, like DDTNs, and that the model as a whole may be a powerful tool for the development of DDTN routing protocols, even in practical scenarios.

The *EG* based protocol has been formalized to provide optimal routing according to its metrics. We implemented the *Foremost Journey* and performed extensive simulations using *NS2*. We compared the performance of the new $EG_{Foremost}$ with four major MANET protocols: DSDV, DSR, AODV and OLSR. The results showed that the drop ratio by no available route (NRTE) using EG was the lowest compared to all protocols in all metrics. Consequently, EG values can be used as a benchmark in many cases.

This first implementation of an *EG* based protocol opens some avenues for further detailed research. For instance, a routing *bottleneck* appears when most used nodes become unavailable for a long period of time, causing overhead when they reappear in the network, leading to packets being dropped due to collision and queue overflows. The use of high values of enforced jitter time when sending packets can minimize the drop rate, but is not feasible in regular protocols. We introduced the SmartJitter as an option to minimize the congestion and achieved good results. The development of a good *EG* adaptive algorithm could possibly manage this problem, anticipating congested nodes in order to find out alternative routing paths. To date, however, there are no theoretical results on flows over *EG*, which is in itself a very interesting open problem.

It should be noted that the high values of average end-to-end delay is an inherent characteristic of the communication network dynamics. In the case of *EG* based protocols, on which the *foremost journey* metric is studied, the end-to-end delay is in any case the minimum arrival date for a packet. If long delays need to be managed, then one policy could be to drop packets that are aged in the network, or – even better – use a *fastest delay* approach, described in [6], instead of the *foremost journey* as done here.

Future work includes implementation of other *EG* based protocols with different metrics, like *shortest path* and *fastest delay*. A natural extension to this work is

related to the deviations in the predicted network dynamics, on which the actual EG used by the nodes is not accurate anymore. This engenders the utilization of a model with stochastically predictable behaviour to better address such variations.

Another open question relates to the case where global knowledge does not exist and it cannot be easily computed locally. In such cases, as suggested by a reviewer, it would be important to address the trade-off which exists between the amount and accuracy of knowledge of network topology and the routing performance. In other words, EG-foremost is a benchmark but its implementation may present a high overhead that is required to achieve complete knowledge of the topology and its evolution; therefore the other routing schemes will be compared as achieving different performances with different routing overheads.

Finally, it would be worth providing a framework that unifies the theoretical power expressed by evolving graphs with the engineering aspects captured in DTNs with knowledge oracles. Major advances in the formalisation of dynamic networks are thus to be expected.

Acknowledgements This work was partially supported by the INRIA-FAPESP project MOBIDYN. Further support from the EU project AEOLUS is acknowledged.

The authors would like to thank Aubin Jarry for his help in the beginning of this work and Olivier Dalle who kindly helped with some of the simulations. The authors are also grateful to the anonymous referees, whose insightful remarks greatly helped increase the quality of the manuscript. The remarks relating *EG*s and DTNs were particularly appreciated.

References

1. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Elsevier) Journal*, 38(4):393–422, Mar 2002.
2. S. Bhadra and A. Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In *Proceedings of Adhoc-Now'03*, volume 2865 of *Lecture Notes in Computer Science*, pages 259–270. Springer Verlag, Oct 2003. Also appeared as an INRIA research report (RR-4531) in Ago. 2002.
3. BonnMotion: A mobility scenario generation and analysis tool. http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/, Page accessed on Oct 2007.
4. A. Boukerche. Performance evaluation of routing protocols for ad hoc wireless networks. *ACM Mobile Network Applications (MONET)*, 9(4):333–342, 2004.
5. J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th ACM annual international Conference on Mobile Computing and Networking (MobiCom'98)*, pages 85–97, Dallas, TX, USA, 1998. ACM Press.
6. B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, Apr 2003. Also appeared as an INRIA research report (RR-4589) in Oct. 2002.
7. C. Carter, S. Yi, and R. Kravets. ARP considered harmful: Manycast transactions in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 03)*, New Orleans, LA, Mar 2003.
8. C. Chen and E. Ekici. A routing protocol for hierarchical leo/meo satellite ip networks. *ACM Wireless Networks (WiNet)*, 11(4):507–521, 2005.
9. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT press, Cambridge, MA, USA, 1990.
10. S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. RFC 2501, IETF, January 1999.
11. DTNRG. Delay tolerant networking research group: http://www.dtnrg.org/. (page accessed on aug. 2007).
12. A. Farago and V. R. Syrotiuk. MERIT: a scalable approach for protocol assessment. *ACM Mobile Networks and Applications (MONET) journal*, 8(5):567–577, 2003.
13. A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, Set 2004. A preliminary version appeared as "On models and algorithms for dynamic communication networks: The case for evolving graphs", Algotel 2002, Mèze, France, May 2002.
14. A. Ferreira, J. Galtier, and P. Penna. Topological design, routing and hand-over in satellite networks. In I. Stojmenovic, editor, *Handbook of Wireless Networks and Mobile Computing*, pages 473–493. John Wiley and Sons, New York, NY, USA, 2002.
15. IEEE standard for wireless LAN medium access control (MAC) and physical layer (PHY) specifications. http://grouper.ieee.org/groups/802/11/main.html, Page accessed on Mar 2007.
16. P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of 5th IEEE INMIC'01*, pages 62–68, Lahore, Pakistan, Dec 2001.
17. S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 145–158, New York, NY, USA, 2004. ACM Press.
18. D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.
19. B. Krishnamachari. *Networking Wireless Sensors*. Cambridge University Press, New York, NY, USA, 2005.
20. B. S. Manoj, K. J. Kumar, C. Frank, and C. S. R. Murthy. On the use of multiple hops in next generation wireless systems. *ACM Wireless Networks (WiNet)*, 12(2):199–221, 2006.
21. S. Merugu, M. Ammar, and E. Zegura. Routing in space and time in networks with predictable mobility. Technical report, Technical Report GIT-CC-04-7, Georgia Institute of Technology, 2004.
22. J. Monteiro, A. Goldman, and A. Ferreira. Performance evaluation of dynamic networks using an evolving graph combinatorial model. In *Proceedings of the 2nd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'06)*, pages 173–180, Montreal, CA, Jun 2006. Best Student Paper Award.
23. NS2. The network simulator – ns2. http://nsnam.isi.edu/nsnam/, Page accessed on Mar 2007.

24. C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Conference on Communications Architectures, Protocols and Applications (ACM SIGCOMM'94)*, pages 234–244, Sep 1994.
25. C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, Feb 1999.
26. C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. In *IEEE Personal Communications*, volume 8, pages 16–28. IEEE Communications Society, feb 2001.
27. Rice University Monarch Project. The CMU monarch wireless and mobility extensions to NS2. http://www.monarch.cs.rice.edu/, Page accessed on Mar 2007.
28. F. J. Ros. UM-OLSR implementation (version 0.8.8) for NS2. http://masimum.dif.um.es/?Software:UM-OLSR, Page accessed on Mar 2007.
29. R. Sen, R. Handorean, G.-C. Roman, and G. Hackmann. Knowledge-driven interactions with services across ad hoc networks. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 222–231, New York, NY, USA, 2004. ACM Press.
30. G. Sharma, R. R. Mazumdar, and N. B. Shroff. On the complexity of scheduling in wireless networks. In *Proceedings of the 12th ACM annual international Conference on Mobile Computing and Networking (MobiCom'06)*, pages 227–238, Sep 2006.
31. I. G. Siqueira, L. B. Ruiz, A. A. F. Loureiro, Jose, and M. Nogueira. Coverage area management for wireless sensor networks. *International Journal of Network Management*, 17(1):17–31, 2007.
32. T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. *to appear in ACM/IEEE Transactions on Networking.*, 2007.
33. I. Stojmenovic, A. Nayak, and J. Kuruvila. Design guidelines for routing protocols in ad hoc and sensor networks with a realistic physical layer. *IEEE Communications Magazine (Ad Hoc and Sensor Networks Series)*, 43(3):101–106, March 2005.
34. I. Stojmenovic(ed.). *Handbook of Wireless Networks and Mobile Computing*. John Wiley and Sons, Feb 2002.
35. I. Stojmenovic(ed.). *Handbook of Sensor Networks: Algorithms and Architectures*. John Wiley and Sons, Oct 2005.
36. J. Wu. *Handbook On Theoretical And Algorithmic Aspects Of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*. Auerbach Publications, Boston, MA, USA, 2005.
37. J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, volume 2, pages 1312–1321, 2003.
38. Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys*, 8(1):24–37, 1st Quarter 2006.