Evolutionary Epidemic Routing

Sara Alouf, Iacopo Carreras, Daniele Miorandi, Giovanni Neglia

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Routage Épidémique Évolutionnaire

Sara Alouf — Iacopo Carreras — Daniele Miorandi — Giovanni Neglia

N° 6140 — version 2

version initiale février 2007 — version révisée mai 2007

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6140--FR+ENG

Routage Épidémique Évolutionnaire

Sara Alouf, Iacopo Carreras*, Daniele Miorandi*, Giovanni Neglia

Thème COM — Systèmes communicants
Projet Maestro

Rapport de recherche n° 6140 — version 2† — version initiale fvrier 2007 — version révisée mai 2007 — 19 pages

Résumé : Nous présentons une approche permettant à des algorithmes de relais d'évoluer afin de s'adapter à un environnement variable et inconnu a priori. Cette approche s'inspire des algorithmes génétiques. Ainsi, la politique de relais utilisée par un nœud est décrite par un génotype; un procédé de sélection promeut la diffusion, parmi les nœuds, des génotypes les plus adaptés et, enfin, de nouveaux génotypes sont créés soit en combinant ceux existants soit en appliquant sur ceux-ci des changements aléatoires. Un cas d'étude illustrant cette approche est présenté et des simulations sont menées afin d'en évaluer les performances.

Mots-clés : Réseaux sans-fil, réseaux à connectivité intermittente, routage épidémique, protocoles évolutifs, algorithmes génétiques

* Create-Net, Italie
† All sections have been updated and a case study with simulation results has been added.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Evolutionary Epidemic Routing

Abstract: In this work, we introduce a framework which allows forwarding schemes to evolve in order to adapt to changing and a priori unknown environments. The framework is inspired by genetic algorithms: at each node a genotype describes the forwarding scheme used, a selection process fosters the diffusion of the fittest genotypes in the system and new genotypes are created by combining existing ones or applying random changes. This framework is illustrated through a simple case study and simulations are undertaken to evaluate its performance.

Key-words: Wireless networks, delay-tolerant networks, epidemic forwarding, evolving protocols, genetic algorithms

1 Introduction

Epidemic-style forwarding [15] has been proposed as an approach for achieving packet delivery in Delay-Tolerant Networks (DTNs) [3], in order to ensure system-wide dissemination of messages in face of frequent disconnections [18, 11]. DTNs are sparse and/or highly mobile wireless ad hoc networks where no continuous connectivity guarantees can be assumed. Epidemic-style forwarding in DTNs is based on a "store-carry-forward" paradigm : a node receiving a message buffers it and carries that message as it moves, passing the message on to new nodes that it encounters. Analogous to the spread of infectious diseases, each time a message-carrying node encounters a new node that does not have a copy of that message, the carrier may decide – according to some specific policies – to *infect* this new node by passing on a message copy ; newly infected nodes, in turn, behave similarly. The destination receives the message when it first meets an infected node.

An unconstrained epidemic forwarding scheme (in which an infected node spreads the epidemic to all nodes it encounters) is able to achieve minimum delivery delay at the expense of an increased use of resources such as buffer space, bandwidth, and transmission power. Variations of epidemic forwarding have been recently proposed in order to exploit the trade-off between delivery delay and resource consumption. This family includes, among the others, K-hop schemes [5], K-copy techniques [2], probabilistic forwarding [8, 6], and spray-and-wait [14, 13]. These schemes differ in their "infection process", i.e., the spreading of a message in network. They need to be combined with a "recovery process" that deletes copies of a message at infected nodes, following the successful delivery of the message to the destination. Various recovery schemes have been proposed : some are simply based on timers, others actively spread in the network the information that a copy has been delivered to the destination, using so-called antipackets [6].

Depending on the specific application scenario, different performance metrics could be envisaged. These include, e.g., the probability to successfully deliver a message to the destination, the delivery time, the total energy consumption in the system or a combination of the previous ones. For a given optimization goal the choice of a specific forwarding scheme and the configuration of its parameters depend in general on the number of nodes in the system, on their mobility patterns and on the traffic generated in the networks [9]. In many scenarios, these characteristics cannot be known at system design and deployment time and can also significantly change across time and space. For example, let us consider a personal digital assistant carried by a user in its daily activities. During the day the node can travel at different speeds (e.g. from zero up to car speed), moving from highly crowded areas (supermarkets, classrooms,...) to less crowded ones, with very different trajectories (straight along a highway or following a random walk from shop to shop) and different levels of power availability.

In order to deal with these issues, various adaptive techniques for message forwarding can be envisaged. This approach is limited in that it requires an *a priori* definition of the actions to be taken to optimize the mechanism for some specific situation. The approach we propose is different. We want to embed the ability to evolve *autonomously* in the forwarding service itself. This is achieved by using concepts and tools from the Genetic Algorithms (GAs) field. Each node employs a (potentially different) forwarding policy, which prescribes the operations to be undertaken when receiving a message destined to another node. Such policy is described by an array of parameters called the

genotype. Genotypes are associated with a fitness measure which, roughly speaking, indicates the ability of the current set of parameters to achieve good performance in the current environment. Fitness is evaluated using local information and feedback which is sent from the destination backwards within ACK messages, which act also as antipackets. When two nodes meet, they may exchange genotypes (and associated fitness levels), updating the pools they maintain. Each node periodically generates a new genotype judiciously using those in its pool and implements the corresponding policy. The whole system is engineered in such a way to present a drift towards higher fitness levels.

The rest of the report is organized as follows. Section 2 overviews the evolutionary delay-tolerant forwarding service engineered in this report, and presents the multiple components of this service : the forwarding policy followed by each node and its unified representation ; the selection process of good forwarding policies and the fitness evaluation process ; and the generation of new policies. A case-study implementation is described in details in Section 3. The outcomes of a simulative study, performed using a freely available software tool, are reported in Section 4. Section 5 concludes the report describing some open research issues.

2 A Framework for Evolution of Forwarding Services

As mentioned in the previous section, in this work we aim to develop a framework that will allow the forwarding service to evolve online in order to optimize a given performance metric and to adapt autonomously to the actual system operating conditions. In order to increase the readability of the report, all the notation employed is summarized in Table 1.

We first observe that multiple forwarding schemes can co-exist at the same time in the network. In fact this form of information delivery, based on the presence of multiple copies in the network, does not need nodes to be compliant with a specific behavior, enhancing system robustness with respect to conventional schemes. This flexibility comes from the completely distributed nature of the forwarding process in epidemic-style relaying, which allows node to use different policies in an uncoordinated fashion.

In our framework, each node can apply a distinct forwarding policy to relay messages to other nodes. We want nodes to *learn* online what is the best forwarding policy (or what are good policies) in the current scenario and change consequently the one they employ. We note that a node cannot evaluate by itself whether its current policy fits the current scenario, because it is in general not aware of the consequences of its actions. For example a given node can never know by itself whether its decisions – according to its forwarding policy – to relay or not to relay a message were the right ones or not. Thus, a node may be relaying a message when the latter has already been delivered to its destination, hence wasting resources. On the other hand, a node may refrain from relaying a message when it happens to be the key node in the message delivery process, e.g., if it is the only node traveling between two disconnected clusters of nodes in the network. It should be clear therefore that only other nodes in the system can evaluate the fitness of a node's forwarding policy. We also observe that the goodness (or fitness) of a node's policy depends on the policies implemented by the other nodes as well. Message delivery is in fact a collaborative process, whose performance depends on the behaviors of all nodes, so that a specific policy can be beneficial or detrimental depending on other nodes actions.

The previous considerations imply the need of an online distributed fitness evaluation process and raises an issue about the use of the fitness in the evolution process. Once a node get marks for its own policy, how should these be used in order to change the policy ? We opt for a blind evolutionary approach, which relies on the following assumption :

Assumption 2.1 *The set of nodes in the network can be partitioned in "large" groups of homogeneous nodes having similar mobility models and traffic patterns.*

If this assumption holds then each node can learn from nodes in the same group : it can make its policy more similar to those policies presenting a higher level of fitness. This suggests that two other components are required in our framework : a unified description of the policies, so that each node can communicate to other nodes the one in use, and mechanisms for the derivation of new (hopefully better) policies from existing ones.

Here we summarize the three fundamental components in our evolutionary framework using genetic algorithms terminology. These components are :

1. the possibility to share with other nodes a description of the specific forwarding mechanism deployed at each node (the *genotype* associated to the forwarding policy employed at the node) ;
2. the possibility for each node to modify its forwarding scheme taking into account the schemes of other nodes (what we call, with a slight abuse of terminology, the genotype *evolution*) ;
3. a consistent process for evaluating the fitness of the schemes employed, rewarding "good" solutions in such a way to foster the diffusion of the fittest forwarding genotypes (performing a sort of *natural selection*).

In the following, we will describe in more detail these three components. But first we briefly describe the differences with respect to conventional genetic algorithms or genetic programming approaches.

2.1 Differences with Genetic Algorithms

A genetic algorithm (or GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems, using techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossing-over [4].

GAs have been successfully employed and showed to perform well over a wide range of optimization problems arising in sciences. It is nonetheless worth recalling that, in general, GAs do not outperform other search techniques (this is a direct consequence of the classical no-free-lunch theorem [16]), but it all depends on the peculiar structure (also referred to as "fitness landscape") of the problem under study. We decided to focus on GAs since they already showed to be able to perform well over commonly found fitness landscapes, while on the other hand presenting a low complexity from an algorithmic perspective.

Let $F(x)$ denote the function to optimize, where x is an abstract representation (called the genotype) of a candidate solution. The fitness of a genotype x_i is defined as $\phi_i := F(x_i)$, i.e. the function to optimize evaluated at $x = x_i$. Genetic algorithms are implemented as a computer simulation in which a population of genotypes evolves toward better solutions, i.e., solutions characterized by a

TAB. 1 – Notation

x_i	genotype of node i
$F(\cdot)$	function to optimize
$f(\cdot)$	performance metric of interest
P_i	forwarding probability in node i genotype
H_i	maximum allowed hop-count in node i genotype
T_D	message delivery time
h	hop-count of the first copy reaching the destination
C_i	number of copies of a given message made by node i
γ	time-equivalent cost of a copy in the cost function
r_i	reward received by node i for spreading a message
ϕ_i	fitness of node i genotype
$\hat{\phi}_i$	estimation of node i genotype fitness
$\hat{\phi}_{i,j}$	estimation of node i genotype fitness known by node j
G_i	set of genotypes in the pool of node i
$p_{i,j}$	probability of selecting genotype i at node j during the reproduction phase
T_g	time between two consecutive reproductions
p_c	crossing-over probability
p_m	bit mutation probability
N	number of mobile nodes
T_s	maximum inter-packet arrival time at each node
L	side size of squared playground
r	transmission range
v	node speed
T_{step}	mobility time step in simulations

higher fitness level. The evolution usually starts from a population of randomly generated individuals and occurs in discrete steps ("generations"). In each generation, the fitness of every individual in the population is first evaluated, then multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. The new population is hence used in the next iterative step of the algorithm. The procedure stops when a "good enough" solution (i.e., one whose fitness level exceeds a given threshold) is found.

In our framework, the population is the set of genotypes present in the system, each genotype representing the forwarding policy of a node in the network. Let us assume there are N nodes and thus N genotypes. Coherently with the above notation, we denote as $F(.)$ the function to be optimized (usually the expectation of a metric of interest), and x_i the genotype used by node i. According to the discussion presented at the begin of Section 2, the performance metric depends, in our case, on all genotypes present in the system, i.e., $F = F(x_1, x_2, \cdots, x_N)$. So a candidate

solution is no longer a single genotype, but rather an array of N genotypes[1]. A simple N-times repetition of the same genotype may well be an optimal solution. However, in general, this will not be case. Since F depends on all genotypes present in the system, we cannot consider ϕ_i, the fitness of genotype x_i, to be the corresponding value of the function to optimize (like in conventional GAs)[2]. We need to have a new definition of the fitness such that fitter genotypes yield better performance according to F. We are going to provide it in Section 2.3.

Another difference in comparison to standard GAs is that many functions $F(\cdot)$ we are interested in (e.g. delivery delay and number of copies) have no closed form in general, but can only be estimated online measuring the performance of the actual running system. This estimation (and hence the estimation of fitness values) will potentially be affected by the randomness inherently present in nodes' mobility, data traffic pattern and wireless channels fluctuations.

Further, in our framework, evolution is an open-ended process, in that the system needs to be able to react to unpredictable changes in the environment. While some of these issues are not new in the community working on evolutionary computing strategies [7], their application in the proposed framework is far from being straightforward.

2.2 The Forwarding Policy

A first step towards an evolutionary forwarding service consists in a formal representation of a generic forwarding scheme. Such descriptions represent the *genotypes* of the implemented forwarding schemes.

A forwarding policy consists of a set of actions to undertake upon message reception. It defines what nodes do when they become within mutual transmission range. The actions can be specified using parameters and may rely on information contained in message headers (like message generation time or count of hops the message has traversed). For instance, a node can transmit a message with probability P as long as the message has not been forwarded more than H times. The values of the two parameters uniquely specify one forwarding policy. A simple binary string can be used to represent the policy as illustrated in Fig. 1, where the genotype spans 12 bits. The 6 leftmost bits represent the numerator a of the fraction $a/63$, which is the forwarding probability value P, while the 6 rightmost bits represent the maximum number of hops H. Hence, out of such representation it is possible to reconstruct the corresponding forwarding policy.

The behavior of a generic forwarding scheme can thus be changed by tuning the values of the parameters that specify the actions to be undertaken.

2.3 The Selection Process

The natural selection process promotes the diffusion of organisms presenting a high fitness level. In much the same way, we want to engineer mechanisms for promoting the diffusion of "good" genotypes already present in the population. Those would be the genotypes yielding good performance

[1] In a homogeneous scenario, all nodes play the same role, hence F does not depend on the order of the genotypes. A candidate solution is just a multiset of N genotypes.

[2] We would have $\phi_1 = \phi_2 = \cdots \phi_N = F(x_1, x_2, \cdots, x_N)$.

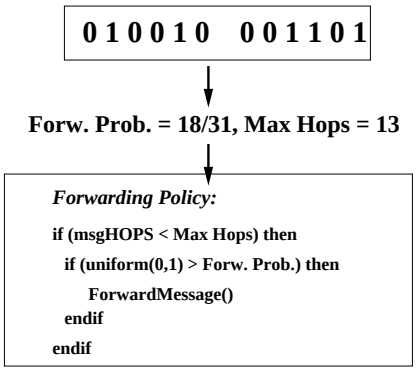

FIG. 1 – Example of forwarding policy mapping : from the binary representation to the forwarding algorithm.

for the optimization goal. At the same time, new genotypes can be generated in order to explore new possible solutions. In this section, we focus on the first issue letting the next section discuss how new genotypes can be generated from existing ones.

In traditional GAs, *reproduction* is a process which selects existing genotypes to create new offsprings. At discrete time intervals, genotypes are randomly selected from the population to generate offsprings : genotype x_i is selected with probability proportional to its fitness, namely, $p_i := \phi_i / \sum_j \phi_j$. This procedure tends by itself to increase the average fitness of the population. However, in the considered mobile network scenario the different genotypes are distributed over all the nodes of the network. This means that standard GAs reproduction phase can not occur without resorting to a centralized solution where a central node (i) stores the genotypes, (ii) applies GA operators, (iii) distribute back the produced offsprings to the mobile nodes. In order to overcome this limitation, we assume that upon meeting nodes exchange information about their respective genotypes and the corresponding fitness indexes. As depicted in Fig. 2, in our system each node maintains a pool of available genotypes (including the one currently in use) and their corresponding fitness values. At regular time intervals, each node goes into a reproduction phase, running the selection process on the genotypes currently stored in its own pool.

We now need to devise a new definition of the fitness (cf. Section 2.1). For the sake of simplicity, we consider that the function to optimize, F, is the expected value of some performance metric, say f, which can be evaluated for a specific infection process. More formally let I refer to the complete history of a generic infection process in the network. An infection I_m reports all of the infection steps since the generation of message m until the cancellation of the message and all its copies. For example it specifies which nodes are infected at a given time instant. The infection I_m depends on the genotypes and the mobility patterns of all nodes involved in the infection process but also on the concurrent data traffic at these nodes. The function that we want to optimize can be rewritten as $F = \mathbb{E}[f(I)]$ where the expectation is taken with respect to the probability measure defined by the mobility and message generation processes. Examples of performance metric $f(I)$ are the time

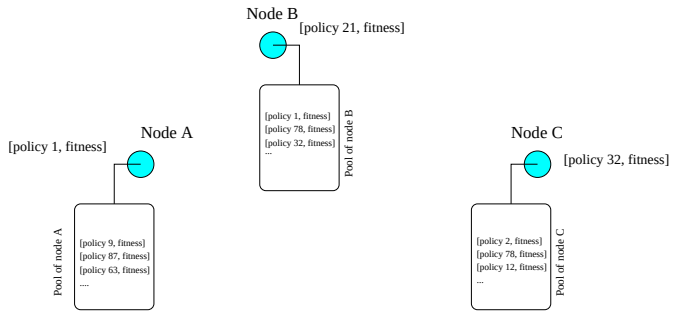

FIG. 2 – Considered system architecture : each node employs a policy, characterized by a genotype with associated fitness value. Each node maintains a pool of candidate solutions used in the reproduction phase to generate new genotypes.

needed to deliver a message to the intended destination, the time before an infection dies (i.e., when all copies are erased from the network), the number of copies done for a given message and the power required to propagate the message.

Observe that an infection I will most likely involve only a subset of the nodes in the system. Conversely, a given node i will take part in only a subset of all infections. The fitness of a genotype x_i can be defined then as

$$\phi_i = \mathbb{E}[f(I) \mid \text{node } i \text{ contributed to infection } I] \ . \tag{1}$$

This ensures that the genotypes of nodes taking part in "good" delivery processes get on average a higher fitness than those involved in "bad" diffusion processes.

According to (1), for a node to estimate the fitness of the genotype it is using, it should average the performance metric $f(I)$ over all infections it had taken part in. A difficulty arises from the fact that the forwarding service is intrinsically a cooperative distributed service and a node is in general not able to evaluate by itself $f(I)$. For example the node does not know whether or not a message has been delivered, when or how many times it has been copied in the system, and so on. Some communication among nodes is thus required in order to let each node be able to evaluate $f(I)$.

In many cases, the process of evaluating $f(I)$ can be triggered by the destination node of a message as it is the best entitled to evaluate the outcome of the infection process. The destination node propagates then the evaluation of $f(I)$ or at least information needed for this evaluation to all nodes involved in the infection process. The communication cost can become significant so that a communication-cost versus information-accuracy trade-off arises. Beside the communication overhead, another aspect to consider is the time needed to evaluate the performance metric $f(I)$. If information is delivered to a node long after message delivery, the node could have changed its genotype, so that the evaluation would not refer to the current genotype.

2.4 Generation of New Forwarding Policies

New forwarding policies are generated applying GA-like operators to the genotypes maintained by a node in its pool. The two following operators can be used to create new genotypes from existing ones. *Crossing-over* consists in breaking two genotypes at a randomly chosen position and exchanging the tails of the genotypes. Two offsprings, called *crossovers*, are produced, and one is selected at random. *Mutation* consists in a random change occurring in the genotypes. As an example, mutation can be implemented by randomly swapping, with some probability, the bits of a binary representation of the genotype. For ensuring stability, mutation should occur with small probability.

2.5 Adaptation vs. Evolution

A possible critic to the framework presented is that it embeds a complex form of adaptation rather than a form of evolution, because the nodes will finally select one genotype out of a finite set. We think that the distinction between adaptation and evolution is in part a problem of scale. Even human DNA has only a finite number of possible different combinations, enabling natural evolution to only choose in a limited set of genotypes. The critical point seems to be the expression capacity of the adopted genotype description. The higher this capacity (and a higher capacity in general requires a more complex language), the larger the number of possible behaviors, perhaps to the extent that the system will start exhibiting some new (unforeseen) behaviors.

3 Mechanism Specifications

In this section we propose a case-study implementation of the proposed approach, aiming at gaining insight into the applicability of the proposed approach to epidemic-style forwarding in delay-tolerant networks. In particular, our purpose is to provide a first answer to the following questions :

(i) Does the distributed genetic algorithm "converge" ?

(ii) If so, what does the convergence point look like and how much time is required for the convergence ?

(iii) What are the performance with respect to an optimally configured static forwarding scheme ?

These questions will be tackled by implementing a (reduced) version of the proposed framework, and running numerical simulations to evaluate, in a realistic scenario, the behavior of the system.

3.1 Implementation of the Evolutionary Process

We consider a simple fixed-length genotype comprising one parameter, which is the probability P to copy a message upon encountering a new node.

Let W denote the set of nodes which contributed to the delivery of the first copy to reach the destination. As optimization goal we consider the minimization of the expectation of the weighted sum of the delivery time, T_D, and the number of message copies made by the nodes in W. The cost

function is then

$$F = \mathbb{E}\left[T_D + \gamma \sum_{i \in W} C_i\right] \quad , \tag{2}$$

where γ is a parameter which can be understood as the time-equivalent cost of a copy.

Should the optimization goal be to minimize solely the expected delivery time, then the evolutionary forwarding scheme will trivially converge, in a underloaded network – i.e., when traffic is small in comparison to the available capacity – to standard epidemic routing, where messages are flooded in the entire network. Conversely, the presence of the number of copies in the cost function makes also an underloaded network (a realistic case and faster to simulate) an interesting scenario to study. The evolutionary forwarding scheme will limit the number of copies depending on the value of the parameter γ. Taking into account the number of copies is also meaningful because they are strongly related to bandwidth usage and power consumption. A more natural choice would be to consider the total number of copies done in the network (i.e. $\sum_{i=1}^{N} C_i$), but a heavier communication overhead would be required. In our case-study we opt for light signaling.

We assume that all nodes are synchronized and that the message header contains a field specifying the time at which the payload was generated at the source[3]. In such a way the destination can evaluate the delivery time as soon as it receives the first copy of a message. On the other hand, each node knows the number of times it has copied a message, but not the delivery time. We assume that each node, before forwarding a copy of a message, adds its own identifier to the message header (this is analogous to what is done in source routing in mobile ad hoc networks). The set W is nothing but the set of node IDs present in the header of the first copy reaching the destination. The destination node sends to nodes in W a new acknowledgment (ACK) message. This message specifies the delivery delay and the number of hops ($h = |W|$) traveled by the message before reaching the intended destination. In this way each node i along the path of the first copy delivered has the following information available : the delivery time, the number of hops in the path and the number of copies it did C_i, the latter value being computed locally at each node. The node evaluates the "reward" obtained for having taken part to the infection, as a decreasing function of the quantity $T_D/h + \gamma C_i$, which we refer to as the "reward" obtained for having taken part to the infection. The fitness of the genotype is then estimated as the average of the rewards obtained. During the selection phase, nodes select genotypes with higher fitness level, so they implicitly select genotypes yielding smaller $T_D/h + \gamma C_i$. The intuition behind this choice is that in this way the system is minimizing[4] :

$$\mathbb{E}\left[\sum_{i \in W} (T_D/h + \gamma C_i)\right] = \mathbb{E}\left[T_D + \gamma \sum_{i \in W} C_i\right],$$

[3]If local clocks are enough accurate at the message delivery time-scale, then there is another solution which does not require synchronization. Message header should have a field which indicates the time since the payload was generated. This field can be updated by each node before forwarding the message. In this case the node should keep track of the time running since it has received the message.

[4] We observe that this is not exact since the number of nodes in W is not constant but it depends itself on the genotypes present in the network.

which is exactly the cost function given in (2). We note that considering the total number of copies would have required each node to be informed about the number of copies done by every other node in the system.

We consider the following expression for the reward at node i :

$$r_i = \max\left\{1 - \frac{T_D/h + \gamma C_i}{R}, 0\right\} \tag{3}$$

where R is set to a high enough value, for instance $2\mathbb{E}[T_D] + \gamma N$.

Upon receiving the n-th ACK message and computing the reward $r_i(n)$ according to Eq. 3, node i updates its estimation of the genotype fitness as follows :

$$\hat{\phi}_i(n) = \frac{n-1}{n}\hat{\phi}_i(n-1) + \frac{1}{n}r_i(n), \tag{4}$$

which corresponds to averaging all rewards received.

When two nodes meet, they transmit to each other their own genotype and its current fitness level estimation. Each node maintains a pool of available genotypes (including the one currently in use) and their fitness. We use G_j to denote the pool at node j and $\hat{\phi}_{i,j}$ to denote the fitness value of node i genotype known by node j ($\hat{\phi}_{i,j}$ is the value of $\hat{\phi}_i$ at the time of the last meeting between node i and node j, so at a given time instant these two values can be different). We consider a synchronized reproduction phase. Every T_g seconds, the *generation lifetime*, nodes synchronously create a new offspring each, i.e., they update their own genotype. This synchronism allows to clearly identify different generations during the evolution. Crossing-over is performed with probability p_c and requires to select two genotypes from the pool, otherwise only one genotype has to be selected. At each node, e.g., at node j, genotypes are selected with a probability proportional to their own fitness[5], namely, $p_{i,j} = \hat{\phi}_{i,j}/(\sum_l \hat{\phi}_{l,j})$ where the sum is over all genotypes contained in the pool G_i. Finally each bit of the current genotype, directly selected from the pool or obtained through a crossing-over of two genotypes, can mutate with probability p_m. The genotype pool is emptied after every generation, and the fitness value of the genotype that will be used is set to zero. If the network is large, the node's pool may not be large enough to keep all genotypes discovered in a generation. Should this be the case, a node may keep only the fittest genotypes, or alternatively select the genotypes according to a fitness-biased distribution.

3.2 Message Structures and Communications

Two nodes are able to exchange messages when they get within mutual communication range. Once it happens, they perform the following steps :

[5]In practice, scaled fitness values are used instead of the raw fitness values $\hat{\phi}_{i,j}$. This is common in GAs and has a twofold purpose. First it avoids having in the first generations few extraordinary genotypes taking over a significant proportion of the finite population in a single generation. Second, should the best fitness values be close to the average ones, more best genotypes than average ones will appear in future generations. We can linearly scale the fitness values such that the best fitness value is double the average fitness value which remains unchanged. In generations where the scaling produces negative normalized fitness values, an alternative scaling is used. The latter maintains equality of the raw and scaled average fitness values, but maps the minimum raw fitness to a null value.

1. exchange node IDs ;
2. exchange header information of data messages ;
3. each node decides which messages should be forwarded to the other node ;
4. messages are exchanged ;
5. each node can drop some messages from its buffer (if full) in order to free space for new messages.

The evolving protocol makes use of two types of messages to be exchanged over the network : DATA messages and ACK messages. DATA messages are those carrying the payload transmitted by any mobile node to a specific destination, whereas ACK messages are used for the following purposes :

- to acknowledge the successful delivery of the message at its intended destination ;
- to feed back the reward to the nodes along the successful path from source to destination (rewarding) ;
- to serve as anti-DATA, by blocking the diffusion of already delivered messages and removing them from nodes buffer.

The fields common to all message headers are (i) [message ID], which is the (unique) identifier (ID) of each message and also specifies whether it is a DATA or an ACK message (ii) [GenTime], which describes the time at which the message has been generated.

Further, data messages include a hop-count field [hops] and the labels of all nodes which forwarded the message along the path from the source to the actual node. ACK messages, on the other hand, include the complete set of nodes involved in the forwarding path and a field, called [Feedback], containing the value of the rewarding metric T_d/h, where T_d is the DATA message delivery time, and h is the value of the field [hops] when the DATA message was received by the destination (i.e., the length of the source-to-destination path).

Each mobile node maintains two internal data structures dedicated to the storage of DATA and ACK messages respectively. In the structure storing DATA messages, each item additionally stores a counter of the number of copies of that message already disseminated in the network.

Whenever a node receives a DATA message to be relayed, it first adds its own node ID to the header and then increments by one the [hops] field of the message. The message is kept in the node's internal memory until the corresponding ACK message is received.

In addition to DATA messages, mobile nodes diffuse also ACK messages. Unlike the case of DATA messages, no limiting policy is applied to the forwarding of these messages (ACK messages are simply ***flooded*** into the network according to the VACCINE recovery scheme [17]).

Whenever it receives an ACK message, a node first adds the received ACK message to the internal message list. It then checks whether the corresponding DATA message is present in its internal memory and, in case, removes it. If the corresponding DATA message was present and the node has contributed to the successful path to the destination (Node ID $\in W = \{$Node ID $1, \ldots,$ Node ID $M\}$), it applies the proper rewarding scheme to update its own fitness, as described in the previous section. The overall procedure is summarized in Algorithm 1.

Algorithm 1 Algorithm performed by a node upon reception of an ACK message.

```
Add the received ACK message to the internal ACK messages list
if msgID ∈ {msgID 1, ..., msgID L} then
    Remove the corresponding DATA message from the internal structure.
    if Node ID ∈ W then
        Update node's fitness value (REWARDING).
    end if
end if
```

TAB. 2 – Simulation Parameters

$L = 500$ m	$T_s = 3000$ s	$T_g = 120000$ s
$r = 25$ m	$T_{\text{step}} = 2$ s	$p_c = 0.10$
$v = 1$ m/s	$\gamma = 100, 400, 800, 1600$ s	$p_m = 0.01$
Static scenario	$N = 20$	
Dynamic scenario	N varies in $\{5, 10, 20, 30, 40\}$	

4 Numerical Results

In order to evaluate the performance of the presented algorithms, we run extensive simulations using the freely available simulation tool OMNeT++ [10].

We consider N mobile nodes, moving at constant speed v over a $L \times L$ square playground according to the random direction mobility model [12]. Each node selects the angular direction of its next movement uniformly in $[0, \pi]$, moves along this direction with a uniform speed ; upon reaching the border, it generates a new angular direction and moves accordingly. The initial locations of nodes are sampled from a uniform distribution which is the stationary distribution of nodes' location under this mobility model (perfect simulation).

Distinct nodes are considered to be in communication range if the mutual distance falls below a threshold r, the communication range. Each mobile node generates a DATA message in a time interval which is uniformly distributed between 0 and T_s seconds, with a destination chosen uniformly among the nodes in the simulation. Each message generated is stored in the out queue of the generating node. The position of every mobile node in the simulation is updated every T_{step} seconds. Each generation lasts for T_g units of time.

The specific values used are in Table 2. Genetic algorithm parameters (p_c, crossover probability, and p_m, mutation probability) are taken to be fixed. The investigation of performance sensitivity to these parameters is left to future work.

As regards the genotype, the forwarding probability P has been quantized non-uniformly using 5 bits for the representation. Such parameter can take values in the set $\left\{(i/31)^{1.5},\ i = 0, 1, \ldots, 31\right\}$.

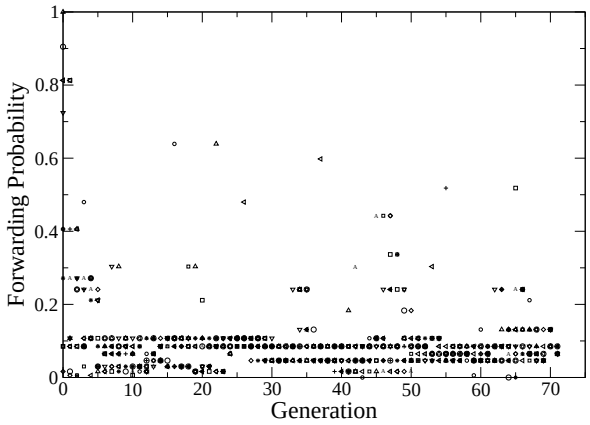

FIG. 3 – Evolution of the forwarding probabilities with $\gamma = 800$.

4.1 Static Scenario

We first consider a case where the number of nodes is kept constant ($N = 20$) throughout the simulation run. The time-equivalent cost of one message copy is set to $\gamma = 800$. The N initial forwarding probabilities are chosen independently in the set of possible values according to a uniform distribution.

Figure 3 depicts the forwarding probability values present in each generation. As it might be seen, after a few generations only small probabilities are used across the population, but for some occasional high values due to random mutations. Simulations with different initial random seeds show similar results.

Figure 4 shows the corresponding evolution over time of the cost (2), expressed in seconds. Observe how the cost rapidly decreases across the first generations and how it "converges" to an almost constant value after 6 generations. Again, running simulations with different initial random seeds yields similar conclusions.

4.2 Comparison with Probabilistic Forwarding

We have conducted a series of simulation runs in which message dissemination was achieved through traditional probabilistic forwarding [8]; that is, a scheme in which all nodes use the same constant forwarding probability. We ran simulations varying the forwarding probability P from 0 to 1 and computed the cost for γ in $\{100, 400, 800, 1600\}$. For each value of γ we conducted simulations using our evolutionary forwarding scheme.

Figure 5 reports the cost, as expressed in (2), achieved by the probabilistic forwarding scheme against the forwarding probability. The cost achieved by our scheme is also reported for the sake of comparison. As it might be expected, for low values of γ, like $\gamma = 100$ (corresponding, roughly

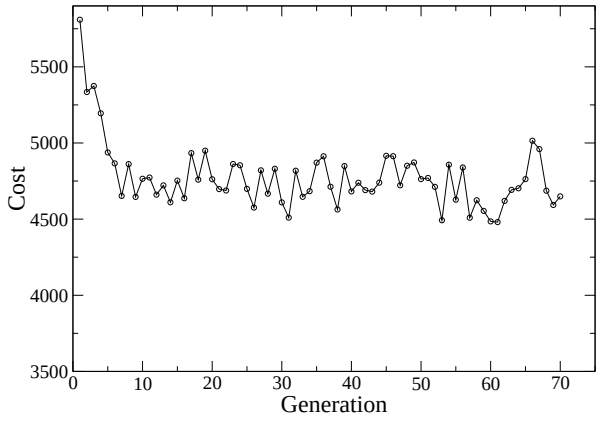

FIG. 4 – Cost averaged over all messages delivered in a generation vs. generation, with $\gamma = 800$.

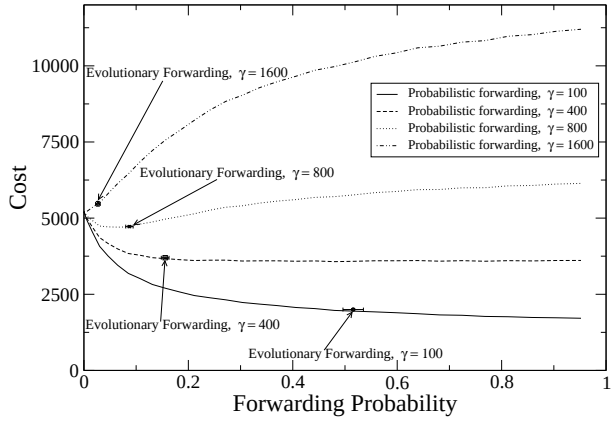

FIG. 5 – Cost vs. forwarding probability for probabilistic forwarding and evolutionary forwarding, $\gamma \in \{100, 400, 800, 1600\}$.

speaking, to a scenario where resources are not an issue but low delays are required), the cost function is monotonically decreasing in P and flooding ($P = 1$) is the best forwarding policy. On the other hand, if the value of γ is high (resource-constrained scenario, e.g. $\gamma = 1600$), the minimum is for $P = 0$ and nodes should make no copy letting the source deliver its message to the destination. For intermediate values of γ (in our case $\gamma = 800$), a minimum exists and a tradeoff between low delay and low resource consumption can be found. The graph reports also the performance achieved by our evolutionary scheme, after the initial convergence transient. The scheme is able to achieve almost optimal performance for all different values of γ.

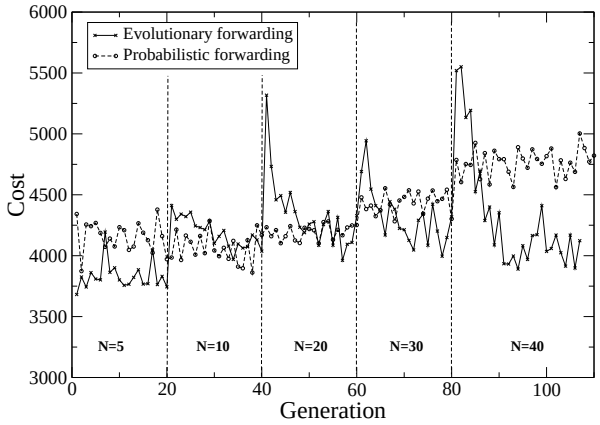

FIG. 6 – Cost vs. time for a dynamic scenario, in which every 20 generations the number of nodes in the network is increased, with $N \in \{5, 10, 20, 30, 40\}$, $\gamma = 800$. The evolutionary forwarding scheme proposed is compared to the traditional probabilistic forwarding scheme with the forwarding probability set to its optimal value when $N = 20$.

4.3 Dynamic Scenario

Last, we consider a simulation case in which N, the number of nodes in the system, varies with time. In particular, it is increased every 20 generations, following the sequence $5, 10, 20, 30, 40$. This dynamic scenario challenges the ability of the proposed framework to track the variations in the network and adapt its parameters accordingly. The time-equivalent cost of a copy is set to $\gamma = 800$. We compute the cost (in seconds) achieved by our evolutionary forwarding scheme in each generation. The results are reported in Fig. 6 which depicts as well the performance of the traditional probabilistic forwarding scheme with the forwarding probability set to its optimal value when $N = 20$ (approximately 0.06, as can be seen from Fig. 5).

As expected, probabilistic forwarding exhibits steady performance as long as the number of nodes does not change. Instead the cost plot of our evolutionary forwarding scheme presents spikes whenever N increases. The abrupt change is mainly due to the arrival of new nodes, whose initial forwarding probability is set at random. During the transient following the spike, genotypes fitter to the new scenario are identified and the cost reduces.

Given a network scenario, our proposed solution exhibits performance similar to a traditional probabilistic forwarding scheme tuned for the specific scenario. But the evolutionary solution outperforms probabilistic forwarding whenever the scenario changes.

5 Outlook and Future Work

In the report, we have presented a framework for embedding autonomous evolution in epidemic-style forwarding schemes. The proposed approach is based on the application of a GA-like mechanism to parameters arrays describing policies employed by the nodes in the system. The simulation results presented indicate in particular that the proposed case-study implementation is able to track changes in the system conditions (e.g., number of nodes in the scenario considered), and achieves similar if not better performance than solutions statically optimized for a given operating point.

The technique proposed in the report represents a first step towards the application of dynamic optimization techniques to forwarding schemes in DTN-like scenarios. In particular, effects of factor like (i) mobility and traffic pattern, (ii) nodes cooperation level, (iii) network heterogeneity, have not been accounted for in the present report, and deserve future studies. Another interesting direction concerns the application of other metaheuristics [1] for run-time optimization.

6 Acknowledgments

The authors would like to thank Lidia Yamamoto for the helpful discussions and the bibliographical references she provided, and Eitan Altman, Konstantin Avrachenkov and Philippe Nain for the interesting brainstorming sessions at early stages of this research.

Références

[1] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization : Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3) :268–308, 2003.

[2] A. Chaintreau, P. Jui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. of IEEE INFOCOM*, Barcelona, Spain, 2006.

[3] K. Fall. A delay-tolerant network architecture for challenged Internets. In *Proc. of ACM SIGCOMM*, Karlsruhe, DE, 2003.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[5] R. Groenevelt, P. Nain, and G. Koole. Message delay in mobile ad hoc networks. *Performance Evaluation*, 62(1-4) :210–228, October 5-7 2005. Proc. of Performance 2005, Juan-les-Pins, France.

[6] Z. J. Haas and T. Small. A new networking model for biological applications of ad hoc sensor networks. *IEEE/ACM Transactions on Networking*, 14(1) :27–40, February 2006.

[7] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3) :303–318, JUN 2005.

[8] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proc. of SAPIR, Fortaleza, Brazil*, volume 3126 of *Lecture Notes in Computer Science*, pages 239–254, August 2004.

[9] G. Neglia and X. Zhang. Optimal delay-power tradeoff in sparse delay tolerant networks : a preliminary study. In *Proc. of ACM SIGCOMM workshop on Challenged Networks (CHANTS)*, September 2006.

[10] OMNeT++ Discrete Event Simulation System. `http://www.omnetpp.org`, 2007.

[11] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking : data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44, November 2006.

[12] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proc. of IEEE ICC*, Helsinki, Finland, June 2001.

[13] T. Small and Z. J. Haas. Resource and performance tradeoffs in delay-tolerant wireless networks. In *Proc. of ACM workshop on Delay Tolerant Networking*, 2005.

[14] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait : an efficient routing scheme for intermittently connected mobile networks. In *Proc. of ACM Workshop on Delay-tolerant networking*, 2005.

[15] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.

[16] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1 :67–82, Apr. 1997.

[17] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. *Computer Networks*, 2007.

[18] Z. Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks : Overview and challenges. *IEEE Comm. Surv. and Tut.*, January 2006.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399