Autonomic Information Diffusion in Intermittently Connected Networks

Sara Alouf, Iacopo Carreras, Álvaro Fialho, Daniele Miorandi, Giovanni Neglia

Autonomic Information Diffusion in Intermittently Connected Networks

Sara Alouf
INRIA, Sophia Antipolis, France, email: sara.alouf@sophia.inria.fr

Iacopo Carreras
CREATE-NET, Trento, Italy, email: iacopo.carreras@create-net.org

Álvaro Roberto Silvestre Fialho
INRIA, Sophia Antipolis, France, email: alvaro.fialho@inria.fr
Now at Microsoft Research-INRIA Joint Centre, Orsay, France

Daniele Miorandi
CREATE-NET, Trento, Italy, email:
daniele.miorandi@create-net.org

Giovanni Neglia
INRIA, Sophia Antipolis, France
University of Palermo, Palermo, Italy, email: tt giovanni.neglia@ieee.org

Abstract

In this work, we introduce a framework for designing autonomic information diffusion mechanisms in intermittently connected wireless networks. Our approach is based on the use of techniques and tools drawn from evolutionary computing research, which enable to embed evolutionary features in epidemic-style forwarding mechanisms. In this way, it is possible to build a system in which information dissemination strategies change at run-time to adapt to the current network conditions in a distributed autonomic fashion. A case study is then introduced, for which design and implementation choices are presented and discussed. Simulation results are reported to validate the ability of the proposed protocol to converge to the optimal operating point (or close to it) in unknown and changing environments.

Keywords: evolving protocols, evolutionary computation, genetic algorithms, wireless networks.

1 Introduction

Epidemic-style forwarding [13] has been proposed as an approach for achieving system-wide dissemination of messages in intermittently connected networks [10] (sometimes named also Delay Tolerant Networks [3]). These networks are sparse and/or highly mobile wireless ad hoc networks where no continuous connectivity guarantees can be assumed. Epidemic-style forwarding is based on a "store-carry-forward" paradigm: a node receiving a message buffers and carries that message as it moves, passing it on to new nodes upon encounter. Alike the spread of infectious diseases, each time a message-carrying node encounters a new node not having a copy thereof, the carrier may decide 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 messages 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, probabilistic forwarding [7], K-hop schemes [4], and spray routing [12].

Depending on the specific application scenario, different performance metrics could be envisaged, such as the probability of successfully delivering 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 its parameters configuration depend in general on the number of nodes in the system, on their mobility patterns and on the traffic generated in the networks [8]. In many scenarios, these characteristics cannot be known at system design and deployment time and may drastically change across time and space.

In order to deal with these issues, various adaptive techniques for message forwarding can be envisaged. Conventional approaches are limited in that they require an *a priori* definition of the actions to be taken to optimize the mechanism for some specific situation. In this chapter, we propose a novel approach, based on the inclusion of autonomic features (in the form of self-optimization capabilities) in the forwarding service itself. This is achieved by using concepts and tools borrowed from the Evolutionary Computation (EC) field. Self-optimization is obtained through a cooperative process, in which nodes exchange their knowledge on the performance of the various schemes employed.

A fundamental observation is that forwarding schemes simply perform a decision whether to relay a copy of a given message to an encountered node or not.

Thus, multiple forwarding schemes can co-exist and interact within the same network. Each node can then employ a potentially different forwarding policy, which prescribes the operations to be undertaken when receiving a message destined to another node. We assume that a way to formally describe forwarding policies is available (it could be as simple as a list of parameters). We call such description the *genotype* of the forwarding policy. 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 local environment with respect to the predefined optimization goal. Fitness estimation is probably the most challenging task in this autonomic service. In fact a node cannot evaluate by itself whether its current policy fits the current scenario, because 1) it is in general not aware of the consequences of its actions (for example a given node may be relaying a message which has already been delivered to its destination) and 2) the fitness of a node's policy depends on the policies implemented by the other nodes as well. Fitness is estimated at each node using not only local information but also feedback from the destination. The fitness estimation process takes into account both feedback delay and feedback noise (due to randomness of the mobility and arrival processes). When two nodes meet, they may exchange genotypes (and associated fitness levels), updating the pool they maintain. Each node periodically generates a new genotype, judiciously using those in its pool. By changing its genotype, a node has made its policy *evolve*. The use of genotypes from other nodes in order to generate a new genotype implements a form of distributed learning, being that each node takes advantage of other nodes' experience. Standard EC operators such as selection, cross-over and mutation are used to generate a new genotype from the pool, but also other approaches are considered. The whole system is engineered in such a way to present a drift towards higher fitness levels. The performance of the proposed scheme is evaluated through extensive simulations, showing its ability to adapt, in a seamless and autonomic way, to varying system conditions.

The remainder of the chapter is organized as follows. Section 2 reviews the related works and motivates our approach. Section 3 describes the autonomic information diffusion service. Section 4 illustrates a case study implementation of the proposed approach to autonomic epidemic-style forwarding in intermittently connected networks. A detailed simulative study, performed using a freely available software tool, provides important insights in the applicability of the proposed approach. Section 5 concludes the chapter.

2 Related Works and Motivation

Intermittently connected wireless systems represent a challenging environment for networking research, due to the problems of ensuring messages delivery in spite of frequent disconnections and random meeting patterns. Due to the mobility of the nodes, protocols such as Ad-hoc On-demand Distance Vector routing (AODV), Dynamic Source Routing (DSR) or Optimized Link State Routing (OLSR), continuously update routes when users require them (AODV and DSR) or in a proactive way (OLSR). These routes commonly time out after a few seconds. When a path between two nodes does not exist through the network, no route can be created. Needless to say that these protocols can hardly run over Delay Tolerant Networks (DTNs) and will fail to deliver data most of the time, because the assumption on the existence at a given time of a complete path between a source and a destination is simply not met.

Many solutions have been proposed for use in such environments over the last few years. The common basis of these solutions is the observation that end-to-end routes may exist over time due to nodes mobility: leveraging their mobility, nodes can exchange and carry other node messages upon meetings, and deliver them afterward to their destinations. This novel routing paradigm is referred to as *store-carry-forward*. Each node in a network serves then as a relay for all other nodes. Different approaches have been proposed depending if contacts among nodes can be planned, predicted, or are unknown in advance. We want to provide a solution for the third case, without relying on any infrastructure or special mobile nodes. We will then briefly review other solutions present in literature for such a scenario.

All of these solutions pertain to the family of epidemic-like forwarding, as they all imitate the spread of viruses in nature. A node having a copy of the message is said to be *infected*, and as such, it can infect any other non-infected nodes that it encounters by passing a message copy to it. If the message is copied at every meeting, the delivery delay is the shortest possible. However, this scheme is extremely wasteful of resources like the channel capacity, the buffer space and the energy. We refer to this protocol as the unconstrained epidemic forwarding scheme.

Vahdat and Becker [13] are the first to propose such a scheme, and call it "epidemic routing". Messages are simply flooded in the network, the only limitation being the maximal number of hops done by a message. The authors show that (i) epidemic routing delivers all packets given unbounded buffer size at each node, and (ii) by appropriately choosing the maximum hop count, delivery rates can still be kept high while reducing resource utilization.

To improve over the simple flooding achieved by epidemic routing, Lindgren, Doria and Schelén propose PROPHET [7], a probabilistic forwarding scheme that is more sophisticated than [13]. Using history of node encounters and transitivity,

PROPHET achieves a performance comparable to that of epidemic routing but with a lower communication overhead.

Groenevelt, Nain and Koole introduce in [4] the multi-copy two-hop relay protocol, a variant of the single-copy two-hop relay protocol proposed and studied in [5]. The infection process is largely constrained as any node can be infected only by the source and can itself infect only the destination.

A new family of routing protocols is proposed in [12] by Spyropoulos, Psounis and Raghavendra. This family, called *Spray routing*, can be viewed as a trade-off between single and multiple copies techniques. Spray routing consists of two phases: the first is called *spray*, and the second is either *wait* (spray-and-wait protocol) or *focus* (spray-and-focus protocol). In the spray phase, a carefully chosen number of copies of the message are generated and disseminated in the network to the same number of relay nodes. In the wait phase, relays simply wait to meet the destination in order to deliver the message. In the focus phase, each copy of the message is routed according to a utility-based single-copy routing algorithm. The authors show that, if carefully designed, spray routing incurs significantly fewer transmissions per message than epidemic routing, and achieves a trade-off between efficient message delivery and low overhead.

Epidemic schemes may be combined with a so-called "recovery process" that deletes copies of a message at infected nodes, following the successful delivery of the message to the destination. Different 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 [6].

In this work, we focus on data dissemination techniques which do not require nodes to exchange any a priori knowledge/estimation of their meeting patterns and/or location. We are interested in epidemic-style forwarding policies due to their simplicity, inherent robustness with respect to node failures and unpredictable system conditions, as well as totally distributed nature. The major drawback of these schemes is that they can potentially harvest the network resources. Also, epidemic-style forwarding is highly sensitive to the setting of some protocol parameters (e.g., the forwarding probability), which in turn should be set according to the current operating conditions. This motivates us to develop an evolutionary forwarding service that will adapt to changes in the network conditions.

3 The Framework

In this work we aim to develop a framework that will allow the forwarding service to evolve on-line in order to optimize a given performance metric and to adapt autonomously to the actual system operating conditions.

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 nodes to use different policies in an uncoordinated fashion.

In our framework, we want nodes to "learn" on-line what is the best forwarding policy (or what are good policies) in the current scenario and change consequently the one they employ. The problem is challenging as 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 a cooperative fitness estimation process has to be put in place in order to allow the network to show self-optimizing features. 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 on-line distributed fitness evaluation process and raise an issue about the use of the fitness in the evolution process. Once a node get marks for its own policy, how should these marks be used in order to change the policy? We opt for a blind evolutionary approach, which relies on a *homogeneity* assumption, according to which the nodes in the network can be partitioned in "large" groups of homogeneous ones, 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 generation of new (hopefully better) policies from existing ones.

Here we summarize the three fundamental components in our evolutionary framework using EC 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. a consistent process for evaluating the *fitness* of the schemes employed, in order to identify "good" solutions;

3. 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*).

In order to enhance the readability, the notation used in this chapter is summarized in Table 1.

3.1 The Forwarding Policy

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

In order to define this formal description, it is essential to identify the key actions potentially involved in a message diffusion process. Whenever two nodes become within each other's transmission range, they have the possibility to exchange some or all of their messages. Regarding the transmission of messages, each of the nodes can perform one of the following actions upon each of its messages: (i) transmitting a copy of the message; (ii) transferring the message itself without keeping a local copy; or (iii) retaining the message. These actions can be further specified and limited by counters and timers or can be performed according to some probability distribution. The messages can be processed on a first come first served basis or according to any other scheduling policy. It is also possible that nodes base their actions on some information contained in message headers (like message generation time or number of hops the message has traversed) and also update it. Regarding the reception of messages, a node receiving a message may face an overflow in its buffer. When this occurs, the node needs to drop some messages from its buffer in order to free space for the new message. Messages to be dropped can be selected at random or according to a specific criterion (the oldest, the most spread, etc.).

All these possible schemes could be specified in different ways, through array of parameters, variable length strings, trees, ... In what follows we briefly describe two possibilities.

Table 1: Notation

x_i	genotype of node i
F	function to optimize
f	performance metric of interest
CF	cost function
P_i	forwarding probability in genotype of node i
H_i	maximum allowed hop-count in genotype of node i
$T_D^{(n)}$	delivery time of message n
$W^{(n)}$	set of nodes in the path of the first copy of message n reaching destination
h_n	hop count of the first copy of message n reaching destination
$C_i^{(n)}$	number of copies of message n done by node i
$C^{(n)}$	total number of copies of a message n in the system
$\hat{C}_i^{(n)}$	estimation at node i of the total number of copies of message n in the system
γ	time-equivalent cost of a copy in the cost function
$R_i^{(n)}$	reward received by node i for spreading message n
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 x_i at node j during the reproduction phase
T_g	time between two consecutive reproductions
p_c	crossing-over probability
p_m	mutation probability
N	number of mobile nodes
T_s	maximum inter-message arrival time at each node
L	side size of squared playground
r	transmission range
v	node speed
T_{step}	mobility time step in simulations

Parameters arrays

We can consider a complex forwarding scheme, which can implement different known schemes, by tuning its parameters. For example the scheme could select the message to forward from the queue according to a probability distribution and copy it only if a local copies counter is below a given threshold and the messages has not traversed more nodes than the value specified by another threshold. This scheme is univocally characterized by its array of parameters including the probability distribution, the copies threshold and the hops threshold. Clearly the scheme could be made more complex and a longer array would be needed.

While this kind of description can become heavy, working with fixed length genotypes simplifies genotype interpretation and genotype processing at each node.

Strings

Another possibility is to rely on strings. The purpose of this section is not to define a formal language but to introduce some considerations and provide some examples.

The genotype string can be divided into two parts: a first part meaningful in transmission mode and specifying the actions to undertake and their parameters when transmitting a message; and a second part meaningful in reception mode and specifying the actions to undertake and their parameters when the buffer is full.

According to the description of nodes operation, we can define many basic genes, which specify actions and their characteristics or parameters. For illustrative purposes we define the following

action genes:
- Select the message (S),
- Transmit a copy of the message (T),
- transFer the message (F),
- Drop the message (D);

parameter genes:
- relaying Nodes counter (N),
- Copies counter (C),
- with probability (P);

and *location* genes:
- uniformly Random (R),
- at the Head (H),
- at the Bottom (B).

Genes (N), (C) and (P) limit the actions of genes (T) and (F) whereas genes (R), (H) and (B) accompany gene (S). If we assume that messages in the queue are chronologically ordered having the most recent one at the bottom of the queue, then genes (H) and (B) respectively represent FIFO and LIFO service disciplines. Note that we could have added a third part to the string representation to define how messages are ordered inside the buffer. However we refrained from doing this so as to lighten the string representation.

As we said we are not going to specify the strings grammar, but we present some possible examples of strings. The purpose is to show how these few genes can yet be sufficient to specify many different behaviors. The two parts of the string will be separated by a dash and genes representing a value will be put between parentheses. Examples are

SBT-SHD: This string describes the standard epidemic routing with priority for recent messages (the node forwards most recent messages and discards least recent ones);

SBTP(0.1)-SRD: This string describes probabilistic forwarding of most recent messages with a forwarding probability equal to 0.1, and where discarded messages are picked uniformly at random;

SRTN(10)-: This string describes a K-hop scheme which limits the diffusion of messages up to the tenth relaying node and where forwarded messages are picked uniformly at random. The absence of the second part could correspond to the case where incoming messages are discarded when the buffer is full (no message already in the buffer would be dropped).

While it does not seem difficult to create a formal language to include such simple examples, the trade-off between the complexity of the description and the variety of possible schemes which can be described (the possible *phenotypes*) has to be investigated.

3.2 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 the genotypes yielding good performance 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 Evolutionary Computation, and in particular in Genetic Algorithms (GAs), *reproduction* is a process which selects existing genotypes to create

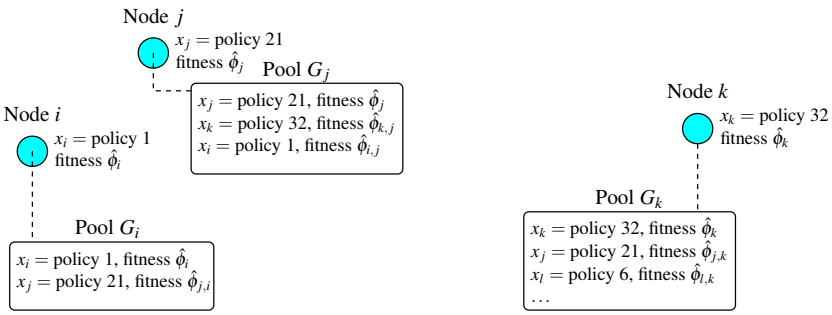


Figure 1: Considered system architecture: each node employs a policy, characterized by a genotype with associated fitness value. Each node maintains a pool of candidate solutions used in the reproduction phase to generate new genotypes.

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, and (iii) distributes back the produced offsprings to the mobile nodes. In order to overcome this limitation and devise a *distributed* solution, we assume that upon meeting nodes exchange information about their respective genotypes and the corresponding fitness indexes. As depicted in Fig. 1, 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.

3.3 Fitness Evaluation

We now need to define formally the fitness of a forwarding policy. 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_n reports all of the infection steps since the generation of message n until the cancellation of the message and all of its copies. For example it specifies which nodes are infected at a given time instant. The infection I_n 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 process, the message generation process. Examples of performance metric $f(I)$ are the time 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 a given infection I will most likely involve only a subset of the nodes in the system. Conversely, a given node j will take part in only a subset of all infections. The fitness of a genotype x_j can be defined then as

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

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 Eq. 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 when or how many times a message is copied in the system, whether or not a message has been delivered, and so on. Some signalling 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. This feedback can be seen as a "reward" to those nodes. 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.

3.4 Generation of New Forwarding Policies

New forwarding policies are generated applying EC-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, or by adding some white noise to the protocol genotype. To ensure stability, mutation should occur with small probability.

4 The Case Study

In this section, we report on a case study implementation of the proposed approach to epidemic-style forwarding in delay-tolerant networks. Our main objective with this implementation is to gain insight into the applicability of the approach proposed in Section 3. In particular, our purpose is to answer 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 is the impact of the mutation process on the speed of convergence?

(iv) How robust is our scheme and how capable is it of adapting to changing network conditions?

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.

4.1 Description

We consider a simple fixed-length genotype comprising one parameter, which is the probability P to copy a message upon encountering a new node. In our prior work [1], a simple binary string was used to represent the value of P. In this work, P is maintained as a double-precision number. As optimization goal we consider the minimization of the expectation of the weighted sum of the delivery time, T_D, and the number of copies of the message done in the system, C. The cost function is then defined as follows

$$CF = \mathbb{E}\left[T_D + \gamma C\right] \quad , \tag{2}$$

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 an 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. Such a metric is also meaningful as it is strongly related to bandwidth usage and power consumption. The evolutionary forwarding scheme will limit the number of copies to an extent that depends on the value of the parameter γ.

Fitness

According to the discussion in Section 3.2, we can think to define the fitness of the genotype at node j as:

$$\phi_j = \mathbb{E}\left[\left(1 - \frac{CF}{R_{max}}\right) \mid \text{node } j \text{ contributed to infection}\right].$$

In this way the fitness is a decreasing function of the cost and if we let R_{max} be a high enough value, we can guarantee that fitness values are positive, as it is required for the biased selection process we described above. We have decided to consider as nodes contributors to the infection of a given message, say message n, only those in the forwarding path of the first copy that reaches the destination. This set of nodes is denoted as $W^{(n)}$. This choice reduces the communication burden in comparison to considering all infected nodes.

Let us discuss now how a node can estimate its fitness. This estimation clearly requires some information about the delivery time and the number of copies done for each message n for which the node is in the set $W^{(n)}$. Regarding the delivery time 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[1]. 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, say message n, adds its own identifier (ID) to the message header (this is analogous to what is done in source routing in mobile ad hoc networks). It follows then that the IDs in the header of the first copy of message n reaching the destination identify exactly the nodes in the set $W^{(n)}$. The destination node sends to these nodes a new acknowledgment (ACK) message.

[1]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.

This message specifies the delivery delay and the number of hops ($h_n = |W^{(n)}|$) traveled by the message before reaching the intended destination.

Estimating the total number of copies is a more complex issue. In the simulation results shown later we assume that nodes know the number of copies done in the system when they receive the ACK and use this number as an estimate of the total number of copies, which will in general be larger being that the infection can still be propagating. We discuss briefly realistic estimation approaches, that we are evaluating. Each node can keep track of the number of copies it did for message n, let us denote as $C_j^{(n)}$ the number of copies done by node j. An approach is then to let each node broadcast its local number of copies and then evaluate the total number of copies aggregating the information received by other nodes ($C^{(n)} = \sum_{j=1}^{N} C_j^{(n)}$), but this would imply a significant communication overhead. We think it is better to rely on an estimation by adopting the "Plain Diffusion" solution proposed by [2]. In this algorithm, two meeting nodes exchange their current estimates and evaluate a new (better) estimate. The estimations at all nodes converge with probability one to the real value, but this requires some time after the end of the infection, hence nodes need to wait after the reception of the ACK to rely on an accurate estimate.

Once a node, say node j, knows the delivery time and has a reliable estimation (say $\hat{C}_j^{(n)}$) of the total number of copies for message n, it can estimate the cost of spreading the message as $T_D^{(n)} + \gamma \hat{C}_j^{(n)}$, a quantity that we refer to as the reward $R_j^{(n)}$, and then update its fitness. Let us denote as M_j the set of all infections node j has been contributing to, formally $M_j = \{n | j \in W^{(n)}\}$. A naive approach to estimate the fitness would be to simply average the rewards over all messages in M_j. Hence we could think to estimate the fitness of node j as: $1 - 1/|M_j| \sum_{n \in M_j} R_j^{(n)} / R_{max}$. In reality this approach would introduce a bias, because infections with longer forwarding paths, would generate a higher number of rewards in the system. The way to eliminate this bias is to consider the following weighted average:

$$\hat{\phi}_j = 1 - \frac{1}{R_{max}} \frac{\sum_{n \in M_j} \frac{R_j^{(n)}}{h_n}}{\sum_{n \in M_j} \frac{1}{h_n}}.$$

Evolution

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 value of node i fitness 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. No cross-over is performed, but only mutation, in the form of addition of a properly defined white noise (described below).

At each node, e.g. at node j, the genotype to be used as a basis for the new generation is selected with a probability proportional to its own fitness[2], 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_j. Finally, one genotype selected from the pool can be mutated with probability p_m, before becoming the active genotype/forwarding policy of the node during the next generation. The genotype pool is emptied after every generation. 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.

As mentioned in Section 3, for stability reasons, a mutation should occur only with some small probability p_m. In [1], we have used a binary representation of the genotype and have implemented mutation by randomly swapping, with some probability, the bits of the binary representation. In this case study, we are considering a *continuous* representation of the genotype (the forwarding probability P). A mutation is performed by summing to the value P a randomly generated "noise". We want the noise to satisfy the following requirements:

1. finer grain for smaller values; this implies (i) a finer grain for negative noise than for positive noise, and (ii) a smaller noise for smaller values;

2. zero mean noise;

3. negative and positive noise that are equally probable.

[2]In practice, scaled fitness values are used instead of the raw fitness values $\hat{\phi}_{i,j}$. The purpose is twofold. First, we want to avoid 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, we would like to have substantially more best genotypes than average ones 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.

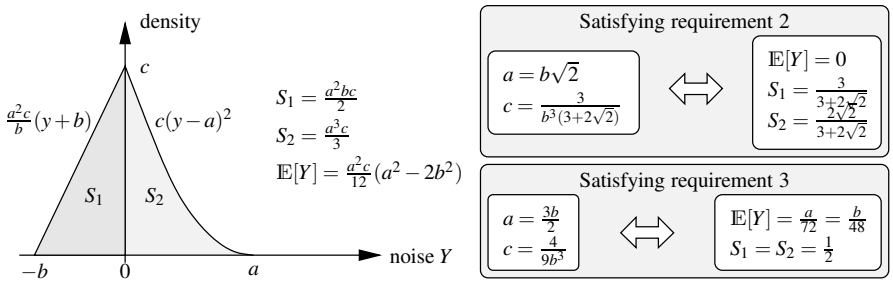


Figure 2: Probability density function of the noise and values of the parameters a, b and c for the satisfaction of requirements 1 and either 2 or 3.

Requirements 2 and 3 are consistent with the well-known additive Gaussian noise, however this type of noise does not satisfy the first requirement. The latter is needed because of the increased sensitiveness of the cost function to lower values of P.

It turns out not to be possible to satisfy all 3 requirements simultaneously. Therefore we have decided for a distribution that satisfies the first two ones.

One simple way of achieving the first requirement is to consider the additive noise to (i) be *proportional* to the value P and (ii) have a distribution with smaller support at negative values. Let Y be a random variable having its probability density function as depicted in Fig. 2. Given that P is the value of the genotype that is selected from the pool, the value of the new genotype is defined as $(1+Y)P$. Clearly, if $\mathbb{E}[Y] = 0$ then $\mathbb{E}[(1+Y)P] = \mathbb{E}[P]$. The density of Y has been defined over the interval $[-b, a]$, where $0 < b < a < 1$. Its particular shape ensures that it is continuous at 0 and that the distribution is as balanced as possible (i.e., $S_1 := P(Y < 0)$ and $S_2 := P(Y > 0)$ as close as possible). To satisfy the second requirement, we let $a = b\sqrt{2}$; see details in Fig. 2, which also shows which values would satisfy the third requirement. In this case, $S_1/S_2 \approx 1.06066$, thus the third requirement is only slightly violated. To satisfy the third requirement, one would let $a = 3b/2$. However, the expected noise will be positive as expressed in Fig. 2. There will then be a bias towards higher values.

From now on, we consider $a = b\sqrt{2}$ and $c = 3/(b^3(3+2\sqrt{2}))$.

One problem is related to the setting of boundaries, as the forwarding probability P should be in the range $[0,1]$. In our case, we have to deal only with the boundary 1, as the genotype can get only infinitely close to 0 but never below 0. Three methods to deal with boundaries are seen in the literature:

1. iterate until the value falls within the boundaries;

2. try once and bounce the value between the boundaries until it falls between

them;

3. try once and truncate the value to the boundary.

All three methods introduce a bias towards smaller values in the distribution of P. The third method is expected to have the smallest bias. In particular, a mutation with a *positive* noise moving the probability beyond the boundary 1 results, with the third method, in a new value (equal to 1) that is *larger* than the original probability. Should any of the first two methods be used, there is no guarantee that a positive noise raising a boundary violation does result in a new value that is larger than the original one.

However, given the structure of the noise, the exploration of the state-space of the probability is much slower towards the left than towards the right. Having then a bias towards smaller values is not as an important issue as one may think at first. Because of the bouncing, the second method is most likely suited only when the noise distribution is symmetric, which is not the case here. Henceforth, we will use the first method to deal with the boundary problem, with a pre-defined (high) limit of trials in order to avoid the system to get trapped on a long sequence of unsuccessful iterations. In case the limit of trials is exceeded, the third method will be applied.

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:

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;

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

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

- 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 the DATA message delivery time T_d. 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. For any DATA message to be relayed, a node 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 internal memory until the corresponding ACK message is received or its lifetime expires.

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 [14]). 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 node has contributed to the successful path to the destination, it updates its own fitness, as described in the previous section.

The overall procedure is summarized in Algorithm 1.

Table 2: Simulation parameters

$L = 2500$m	$T_g = 1500000$s	$\gamma = 200, 1000, 1800$s
$r = 25$m	$T_s = 10000$s	$p_m = 0.1, 0.2$
$v = 1$m/s	$T_{\text{step}} = 2$s	$b = 0.25, 0.5$
Static scenario	$N = 20, 100, 500$	
Dynamic scenario	N varies in $\{20, 60, 80\}$	

4.2 Performance Evaluation

In order to evaluate the performance of the presented algorithms, we have run extensive simulations using the freely available simulation tool OMNeT++ [9].

We consider N mobile nodes, moving at constant speed v over a $L \times L$ square playground according to the random direction mobility model [11]. 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. Nodes initial locations are sampled from a uniform distribution which is the nodes stationary distribution under this mobility model (perfect simulation).

Distinct nodes are considered to be in communication range if the mutual distance falls below the communication range r. 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. The default values of the mutation process are $p_m = 0.01$, and $b = 0.25$ ($a = b\sqrt{2}$, $c = 3/(b^3(3+2\sqrt{2}))$; see Fig. 2).

4.2.1 System Convergence

Our first objective is to assess the ability of the evolutionary forwarding mechanism to reach the optimal operating point, which is represented by the minimum of the cost function across all the possible values of the forwarding probability.

We have thus conducted a series of simulation runs with $N = 100$ nodes in which message delivery was achieved through pure probabilistic forwarding, with all nodes applying the same fixed forwarding probability. We ran simulations varying the forwarding probability P from 0 to 1 and computed the cost for $\gamma \in \{200, 1000, 1800\}$. For each γ we conducted simulations using our evolutionary

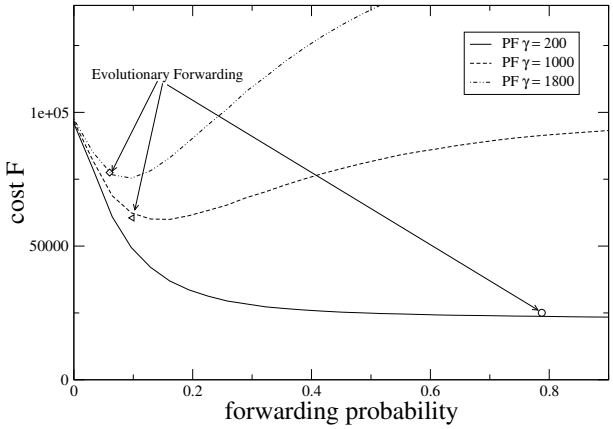
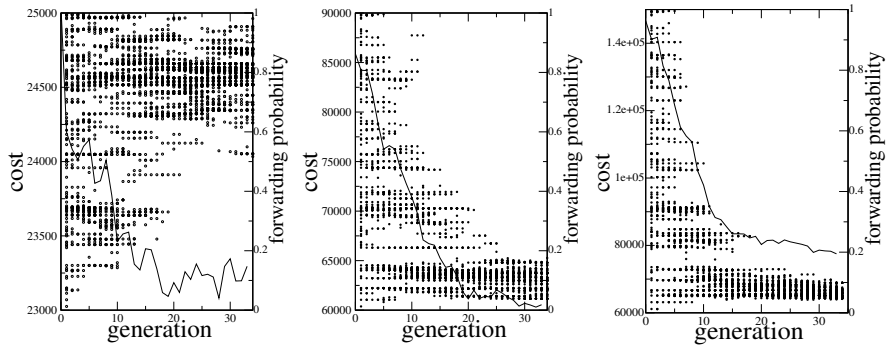


Figure 3: Cost with probabilistic forwarding (PF) and evolutionary forwarding.

forwarding scheme. Nodes genotypes are initialized by selecting forwarding probabilities uniformly at random in the interval $[0,1]$.

In Fig. 3 each curve shows the cost, as expressed in (2), achieved by the probabilistic forwarding scheme as the forwarding probability changes, for a specific value of the parameter γ. As one could expect, for low values of γ, like $\gamma = 200$ (corresponding, roughly 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. As γ increases, the number of message copies done in the system becomes increasingly important, and naturally, smaller forwarding probabilities start achieving lower cost values. For $\gamma = 1000$ and $\gamma = 1800$, a minimum exists and a trade-off between low delay and low resource consumption can be found. Three dots represent the performance of our scheme after the initial transient: the abscissas are obtained averaging forwarding probability values across all the nodes. As it can be easily seen, the system running our scheme is able to reach, after convergence, an operating point that is very close to the optimal one, for the three values of γ that were considered.

In the proposed system, evolution occurs at regular intervals, the generation period. Hence, at each generation instant an evolution occurs, and the system changes its behavior updating the forwarding probability toward those values minimizing the cost function. This effect is evident in Fig. 4, which depicts the distribution of the forwarding probability and the cost function CF over the generation number. As it might be seen, after approximately 20 generations, the forwarding probabilities used in the system converge. The average forwarding probability values after convergence are the ones that are reported on each curve of Fig. 3. The system



(a) $\gamma = 200$ (b) $\gamma = 1000$ (c) $\gamma = 1800$

Figure 4: Cost function and genotypes evolution over time when $N = 100$ nodes.

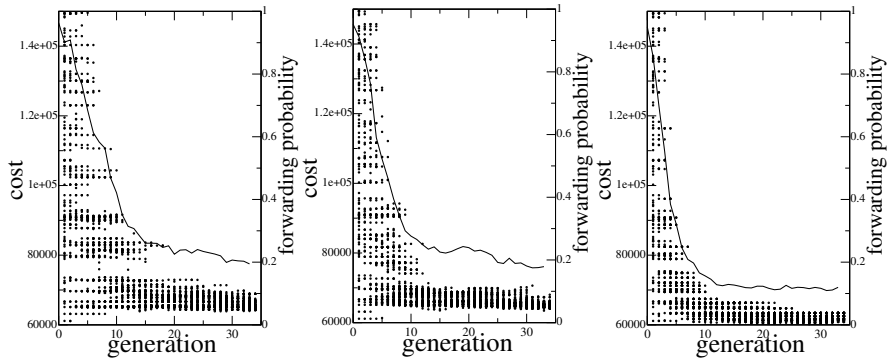
does converge around the optimal operating point.

Impact of the Mutation Process

We have then analyzed the impact of different evolution parameters, i.e., mutation probability, noise settings. In Fig. 5, we report the distribution of the forwarding probability, together with the cost function, over generations. Figure 5(a) reports the performance under our default mutation settings. In Fig. 5(b), we have kept the same distribution of noise (process Y in Section 4) but have considered a larger mutation probability $p_m = 0.2$. The convergence of the cost function appears to be a couple of generations faster. In Fig. 5(c), we have kept $p_m = 0.1$ but have considered a process Y with a larger variance (fourfold the one of the default settings). Interestingly, the system converges much faster than with the default settings for either $p_m = 0.1$ or 0.2. The convergence seems to occur around the 10^{th} generation. Also, the variance of the noise has a larger impact on the convergence than the probability of mutation.

4.2.2 System Scalability

We have next considered the scaling properties of the proposed evolutionary mechanism. We repeated the same steps described in Section 4.2.1 and whose results are reported in Fig. 3, but this time with $N = 20$ and $N = 500$. The results are reported in Fig. 6. Figure 6(a) (resp. 6(b)) depicts the cost function versus the forwarding probability when a fixed probabilistic forwarding is used, with $N = 20$ nodes (resp. $N = 500$ nodes). The operating point to which our evolutionary scheme converges



(a) $p_m = 0.1, b = 1/4$ (b) $p_m = 0.2, b = 1/4$ (c) $p_m = 0.1, b = 1/2$

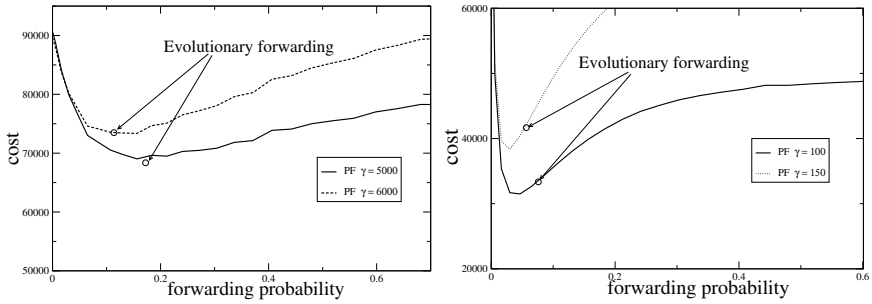
Figure 5: Impact of the mutation process ($N = 100$ nodes, $\gamma = 1800$).

is also reported. We can say that the system, when running our scheme, reaches an operating point that is close to the optimal one for a large range of network sizes (N ranging from 20 to 500). We believe that our scheme should scale to even larger network sizes. Unfortunately, fine-grained simulations do not scale as easily to large networks.

4.2.3 System Robustness

The last point that we wanted to address through simulations concerns the robustness of our scheme and its ability to adapt to a changing environment. Therefore, we considered a scenario in which the number of nodes changes abruptly after 20 generations from 10 nodes, up to 100 nodes, then up to 300 nodes after other 20 generations. It then reduces to 100 at the 80th generation and finally reduces back to 10 nodes at the 100th generation. The time-equivalent cost of a copy is set to $\gamma = 1800$ and the noise parameter b is set to 0.5.

Figure 7 depicts, over the generations, the cost achieved by our scheme and the average forwarding probability among all nodes. The cost plot presents spikes whenever N increases. The abrupt change is mainly due to the arrival of new nodes, whose initial forwarding probability is set uniformly at random. During the transient following the spike, genotypes better fitted to the new scenario are identified and the cost reduces. As an example, the system is able to let the forwarding probability decrease when the number of nodes increases. However, a similar behavior is not observed in the opposite case – when the number of nodes decreases. This is rooted in the limited *diversity* of the nodes' genotypes. When adding nodes, the newcomers take a forwarding probability chosen at random. This randomness in-



(a) 20 nodes (b) 500 nodes

Figure 6: Convergence of the evolutionary forwarding scheme in the case of 20 and 500 nodes.

jects a sufficient level of genotype diversity that allows our scheme to explore, in parallel, a wide range of forwarding probabilities. Conversely, once the system has converged to the optimal value for a specific setting and the scenario changes, the system is extremely slow in its reaction due to a limited genetic diversity. This is evidently clear in Fig. 7, where our scheme is not able to track the changes deriving from the removal of nodes.

4.3 Alternative Approaches

In our scheme we rely on standard EC techniques to select genotypes from the pool at each node and to generate new ones. In reality, as Fig. 3 shows, the fitness landscape in our networking problem is quite regular, hence we can think to take advantage of this.

A possibility is to use gradient descent-like techniques. The genotype-fitness pairs in the pool of a node can be considered as samples of the whole fitness landscape. These samples can be used to estimate the gradient of the fitness curve in correspondence of the genotype used by the node and then change it moving *nearer* to the minimum. Fitness samples are noisy, hence the gradient has to be carefully evaluated. At the same time taking into account the intrinsic correlation of the fitness values for similar genotypes should reduce the effect of noise: e.g. an anomalous fitness estimate could lead many nodes to erroneously select that genotype if standard biased selection is applied, but it should not alter significantly gradient calculations.

This idea can be further developed. In fact in our scenario, a node is not constrained to a local gradient evaluation which relies only on fitness estimates for

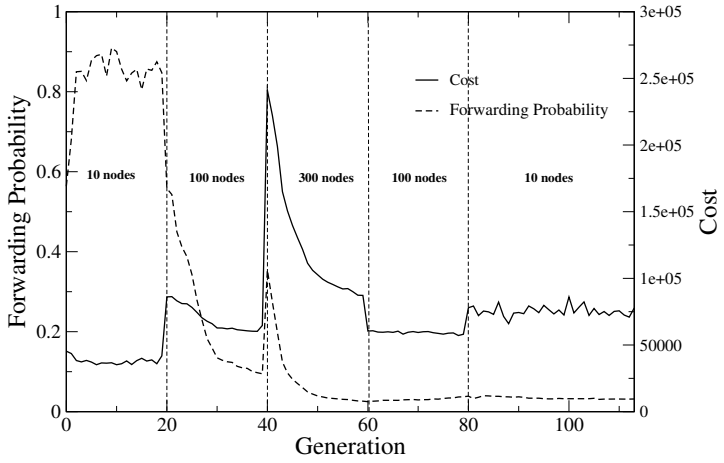


Figure 7: Dynamic scenario: cost function and forwarding probabilities over the generations.

near genotypes, but it can exploit all the genotypes in its pool, which are in general spread across all the genotype space. The minimum of the function can be directly estimated using all the fitness values, hence further reducing the effect of noise. For example, a least squares fitting can be applied, the minimum can then be evaluated on the basis of the estimated curve, and the node genotype can be made closer to this minimum. This approach should also be integrated with a random mutation, otherwise after some time all genotypes would converge to the same value.

Another possible approach is to renounce to genotype exchange and let each node learn autonomously relying only on its past experience. The same mechanisms described up to now can be applied if we think about the pool as a collection of genotypes and related fitnesses as evaluated in the past by the node itself (and not by different nodes). This solution would definitely make the system more robust to mal-functioning or malicious nodes, which can currently influence genotype selection by other nodes, just by spreading false fitness values, but at the same time longer convergence time would be expected.

5 Conclusion

In this chapter, we have presented a framework to learn in a distributed and on-line way, a good forwarding policy in delay tolerant networks. The most challenging aspect of this framework concerns the estimation of the fitness of the genotype

used at a node. Each node contributes to maximizing a global objective function using local knowledge. We have proposed a case study to illustrate the framework. Extensive simulations have illustrated (i) the convergence of the system even when the network counts as much as 500 nodes, (ii) the impact of the mutation process onto the convergence of the system, and (iii) the capability of the system to adapt to changing network conditions.

References

[1] Alouf S, Carreras I, Miorandi D, Neglia G (2007) Embedding evolution in epidemic-style forwarding. In: Proc. of IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS 2007), Pisa, Italy

[2] Babaoglu O, Canright G, Deutsch A, Caro GAD, Ducatelle F, Gambardella LM, Ganguly N, Jelasity M, Montemanni R, Montresor A, Urnes T (2006) Design patterns from biology for distributed computing. ACM Trans Auton Adapt Syst 1(1):26–66

[3] Fall K (2003) A delay-tolerant network architecture for challenged Internets. In: Proc. of ACM SIGCOMM 2003, ACM, New York, NY, USA, pp 27–34

[4] Groenevelt R, Nain P, Koole G (2005) The message delay in mobile ad hoc networks. Performance Evaluation 62(1-4):210–228

[5] Grossglauser M, Tse D (2002) Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Trans on Networking 10(4):477–486

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

[7] Lindgren A, Doria A, Schelen O (2004) Probabilistic routing in intermittently connected networks. In: Proc. of SAPIR Workshop 2004, LNCS, vol 3126, pp 239–254

[8] Neglia G, Zhang X (2006) Optimal delay-power tradeoff in sparse delay tolerant networks: a preliminary study. In: Proc. of ACM SIGCOMM CHANTS 2006, pp 237–244

[9] OMNeT (2007) OMNeT++ Discrete Event Simulation System. `http://www.omnetpp.org`

[10] Pelusi L, Passarella A, Conti M (2006) Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. IEEE Comm Mag 44(11):134–141

[11] Royer EM, Melliar-Smith PM, Moser LE (2001) An analysis of the optimum node density for ad hoc mobile networks. In: Proc. of IEEE ICC 2001, vol 3, pp 857–861

[12] Spyropoulos T, Psounis K, Raghavendra CS (2008) Efficient routing in intermittently connected mobile networks: The multiple-copy case. IEEE/ACM Trans on Networking 16(1):77–90

[13] Vahdat A, Becker D (2000) Epidemic routing for partially connected ad hoc networks. Tech. Rep. CS-200006, Duke Univ.

[14] Zhang X, Neglia G, Kurose J, Towsley D (2007) Performance modeling of epidemic routing. Computer Networks 51(10):2867–2891