Exploring the Interdisciplinary Connections of Gossip-based Systems*

Paolo Costa
Vrije Universiteit, Amsterdam
costa@cs.vu.nl

Vincent Gramoli
INRIA, Université Rennes 1
vgramoli@irisa.fr

Márk Jelasity
University of Szeged and HAS
jelasity@inf.u-szeged.hu

Gian Paolo Jesi
University of Bologna
jesi@cs.unibo.it

Erwan Le Merrer
FTR&D/IRISA, France
elemerre@irisa.fr

Alberto Montresor
University of Trento
montreso@dit.unitn.it

Leonardo Querzoni
University of Rome "La Sapienza"
querzoni@dis.uniroma1.it

ABSTRACT

In recent years the labels "gossip" and "gossip-based" have been applied to an increasingly general class of algorithms, including approaches to information aggregation, overlay network management and clock synchronization. These algorithms are intuitively similar, irrespective of their purpose. Their distinctive features include relying on local information, being round-based and relatively simple, and having a bounded information transmission and processing complexity in each round. Our position is that this class can and should be significantly extended to involve algorithms from other disciplines that share the same or similar distinctive features, like certain parallel numerical algorithms, routing protocols, bio-inspired algorithms and cellular automata, to name but a few. Such a broader perspective would allow us to import knowledge and tools to design and understand gossip-based distributed systems, and we could also export accumulated knowledge to re-interpret some of the problems in other disciplines, such as vehicular traffic control. In this position paper we describe a number of areas that show parallels with gossip protocols. These example areas will hopefully serve as inspiration for future research. In addition, we believe that comparisons with other fields also helps clarify the definition of gossip protocols and represents a necessary first step towards an eventual formal definition.

1. INTRODUCTION

1.1 What makes gossip different?

Since the seminal paper of Demers et al. [12] the idea of epidemic (or gossip) algorithms has gained considerable popularity within the distributed systems and algorithms communities. The gossip approach to information dissemination represents a departure from the mainstream of traditional distributed algorithms. In their case, researchers typically start with desirable properties, then they design algorithms with exact provable guarantees for these properties in various system models.

In contrast, gossip protocols are inherently probabilistic. Nodes perform a simple set of operations periodically. They are not aware of the state of the entire system, only a very small fraction of it, and act based on solely local knowledge. Yet, in a probabilistic sense, the system as a whole achieves very high levels of robustness to benign failures and a favorable (typically logarithmic) convergence time.

At an intuitive level, a key feature is that these favorable global properties seem to come for "free", in the sense that the local algorithm running at each node is extremely simple that makes no reference to these properties in any way. In many cases gossip protocols might even display global properties that are unexpected, that is, that were not thought of in the design phase. By definition, these properties are *emergent*. It is not uncommon that new results on global properties of well-known gossip protocols appear well after the first publication of the algorithm itself. In some cases, useful global properties are known only through experimentation without much useful theory.

1.2 A semantic shift in the term "gossip"

In the past two decades, a number of protocols have been proposed that share some key intuitive properties of gossip, and can be considered "emergent" in the loose sense described above. Due to their similar communication and algorithmic structure, and their similar convergence properties, these protocols have also been called gossip or gossip-based protocols. Examples include aggregation [33, 35, 48] and overlay network construction [31, 49].

However, these protocols are clearly not gossip in the strict sense that implies information dissemination. Instead, the term gossip has undergone a semantic shift and has come to signify a certain class of protocols that share the most salient and discriminating features of the original gossip approaches.

1.3 Our position

In the light of the observations above, we can now formulate our positions.

1. **Definition of gossip needs to be revisited.** Due to the semantic shift in the meaning of the term gossip, it is no longer clear what protocols we should call gossip, and what it is that is common to all protocols that we currently do call gossip. Clarifying this problem is

*Authors are listed in alphabetical order

```
1: loop
2:    wait(Δ)
3:    p ← selectPeer()
4:    send state to p
5:    receive state_p from p
6:    state ← update(state_p)
7: end loop
```

(a) active thread

```
1: loop
2:    receive state_q from q
3:    send state to q
4:    state ← update(state_q)
5: end loop
```

(b) passive thread

Figure 1: The gossip protocol.

important because without clear language it is difficult to identify the important problems and possibilities of the field, and professional communication about gossip protocols looses efficiency as well.

2. **Before a formal definition, gossip needs to be clarified informally.** As we will elaborate in Section 2, a definition based on a set of strict formal criteria without a firm intuition is likely to be both too general and too specific: it is likely to cover algorithms that do not "feel like" gossip, yet it can easily exclude many protocols that we do consider gossip. Adopting an informal, prototype-based definition as a first step, focusing on examples, we hope to be able to make progress towards an eventual more formal approach.

3. **Other disciplines study algorithms that we would consider gossip.** We will present detailed examples in Sections 3 and 4.

4. **We can potentially transfer ideas and tools among these disciplines.** In other words, if we identify another discipline as falling into the gossip class of protocols, we can potentially use some of the tools and results the other discipline accumulated to design better or new distributed algorithms and systems, or understand existing ones better.

2. PROTOTYPICAL GOSSIP

According to our position described above, we will avoid a formal definition of gossip in this position paper, and instead we will follow a prototype-based approach. In other words, we will describe a set of properties that we consider characteristic of gossip algorithms, and we will examine a number of examples for these properties.

To illustrate the potential problems with formal definitions, consider the protocol skeleton in Figure 1. This skeleton is often used to define the set of gossip protocols, where one can implement the three main hooks: selectPeer, state, and update, to instantiate the framework.

For example, in the case of anti-entropy gossip for database update spreading, selectPeer returns a random peer from the network, the state is the set of updates a peer is aware of, and update merges the new updates in the received state into the local state. Other protocols can easily be described a similar way.

However, this framework is too general. Indeed, it covers practically all message passing protocols. For example, since the definition of state is unrestricted, in any round a peer can choose to send a zero length message (that is, no message). Besides, the frequency of the rounds can be arbitrarily fast. Furthermore, selectPeer can be deterministic, the messages can be very large and method update can perform arbitrarily complex operations.

If we attempt to add further restrictions in order to arrive at a more meaningful definition—such as requiring that selectPeer is uniform random, or that a protocol must actually send a message in each round—we obviously exclude protocols that are clearly gossip. For example, many secure gossip protocols in fact use deterministic peer selection on controlled networks [34], and, quite clearly, a protocol remains gossip if message sending is slightly irregular—for example, due to an optimization that makes a protocol adaptive to system load or the progress of information spreading.

For this reason it appears to be more productive to work with a feature list that defines an idealized prototypical gossip protocol, and compare the features of a given protocol with this set. In this way, instead of giving a formal, exact definition of gossip protocols, we make it possible to compare any given protocol to the prototypical gossip protocol and assess the similarities and differences, avoiding a rigid binary (gossip/non-gossip) decision over protocols. A flexible approach like this is especially useful for our present purpose of identifying related interdisciplinary fields. We propose the following features:

1. **random peer selection**
2. **only local information is available at all peers**
3. **round-based (periodic)**
4. **limited transmission and processing capacity per round**
5. **all peers run the same algorithm**

We should stress here that this list is not meant to be exclusive to gossip protocols: many (some would argue, most) protocols are round-based, local, etc. In addition, these features are intentionally left fuzzy: for example "limited", "local" or "random" is not defined any further.

The inherent and intentional fuzziness in this prototype-based approach turns the yes/no distinction of a formal definition into a measure of distance from prototypical gossip: a certain algorithm might have some of the properties, and might not have some others. Even in the case of matching properties, we can talk about the degree of matching. For example, we can ask how random peer selection is, or how local the decisions are.

Figure 2 is a simple illustration of this idea. The figure also illustrates the possibility that some algorithms from other fields might actually be closer to prototypical gossip than some protocols currently called gossip.

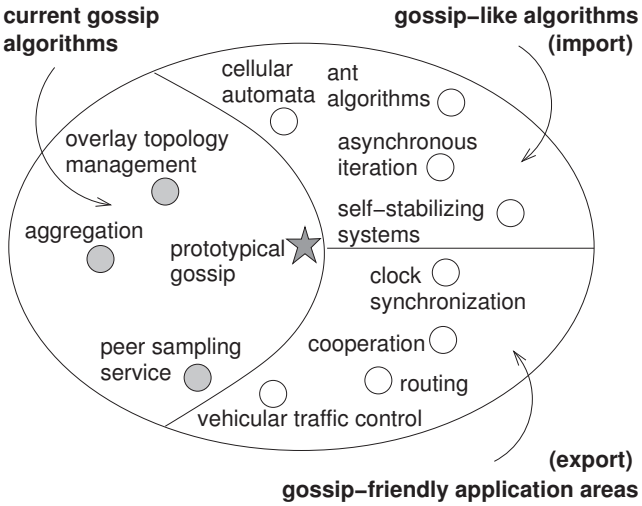

Figure 2: Prototypical gossip placed in a multidisciplinary context through examples discussed in this paper.

One could argue that prototypical features include scalability and robustness to benign failure. However, these features are not the function of the algorithm alone, but, for example, the function of the underlying communication network as well. A perfectly prototypical gossip protocol becomes unscalable when put onto a wireless ad hoc network, or on a physical network with a star topology, and it becomes fragile too, when run on an underlying network with bottleneck links or nodes.

We do not want to mix issues related to application layer gossiping and the underlying physical network. The interaction of these two components is a very interesting and important topic, but it is outside the scope of the present discussion.

3. IMPORTING IDEAS

In this section we will discuss four areas that show significant similarities to prototypical gossip as described above, and outline what possible knowledge transfer might be possible.

The areas discussed here should by no means be interpreted as an exclusive list. Many other fields could have been added, including evolutionary computing, certain multi-agent systems, and so on. The examples were selected to illustrate our position that a multidisciplinary approach might be useful and productive.

3.1 Asynchronous Numerical Iteration

A traditional application in parallel computing involves operations on large matrices and vectors. The example that we will look at is the problem of calculating the dominant eigenvector of a large sparse matrix. For matrix A, an eigenvector of A is any vector $\mathbf{x}$ for which $A\mathbf{x} = \lambda\mathbf{x}$, where λ is the eigenvalue that belongs to $\mathbf{x}$. The dominant (or principal) eigenvector is the one that belongs to the eigenvalue with the largest absolute value. Eigenvectors and eigenvalues have a large number of applications, the most well known within the field of distributed computing probably being the

1: let $\mathbf{x}^{(0)}$ be a random vector ($\|\mathbf{x}^{(0)}\| = 1$)
2: **for** $i = 1, 2, \ldots$ **do**
3: $\quad\mathbf{y} \leftarrow A\mathbf{x}^{(i-1)}$
4: $\quad\mathbf{x}^{(i)} \leftarrow \mathbf{y}/\|\mathbf{y}\|$
5: $\quad$**if** $|1 - {\mathbf{x}^{(i)}}^T\mathbf{x}^{(i-1)}| < \epsilon$ **then**
6: $\quad\quad$**return** $\mathbf{x}^{(i)}$
7: $\quad$**end if**
8: **end for**

Figure 3: Power method

PageRank algorithm [42].

In our example, the elements of the vector are held by individual network nodes, one vector element per one node. The matrix is represented by *links* between the network nodes, and *weights* assigned to these links. In networking terms, the links between the nodes can be physical or virtual (overlay or application level).

For a matrix A, the element A_{ij} is represented by a link from node j to node i with the assigned weight of A_{ij}. If there is no link from j to i then $A_{ij} = 0$.

The dominant eigenvector can be calculated using a number of iterative methods, the simplest being the power method [4]. The algorithm of the power method is shown in Figure 3. To implement the power method, the matrix needs to be accessed only in the form of a vector multiplication, that is, we need to be able to calculate the product $A\mathbf{x}$ for a vector $\mathbf{x}$. The only exception is line 4, where we normalize for length. This step can be omitted if the dominant eigenvalue is 1 or -1. For simplicity, we will assume this from now on.

Let the element $w_i^{(k)}$ of the approximation of the dominant eigenvector in iteration k be stored at node i. With matrix multiplication $\mathbf{w}^{(k+1)} = A\mathbf{w}^{(k)}$, we get

$$w_i^{(k+1)} = \sum_j A_{ij} w_j^{(k)}, \tag{1}$$

which can be computed if all nodes "propagate weights" through the links starting from them, that is, they multiply their own value by the weight of the outgoing link, and send a message to the target of the link containing this product. All nodes sum up the weights they receive and this gives the new vector elements.

This, if done in lockstep by all nodes, implements the power method. Note the apparent similarity with the gossip properties: nodes only use local information, the processing and bandwidth utilization is limited if the matrix is sparse (that is, if all nodes have a limited number of neighbors), it is round-based, and finally, all peers run the same protocol.

The only issue is peer selection: in this version of the protocol all neighbors are contacted in each round. These neighbors are not selected at random but fixed throughout the computation. However, randomness can be introduced without hurting the convergence properties. A version of this algorithm was proposed [38] that converges to the right solution even if processes do not work in lockstep, neighbors are contacted at random, and messages can be delayed or lost. With this algorithm, peers can be selected at random from the fixed set of neighbors, making this iteration protocol very close to our prototypical gossip protocol, even for non-sparse matrices.

This striking similarity suggests that perhaps some gossip

protocols can be reinterpreted as numeric iterations, which would make it possible to make use of a wide array of theoretical results and tools connected with convergence and fault tolerance.

3.2 Ant-based Protocols

Ant protocols are a wide family of protocols which find inspiration in the behavior of real ants. More than fifty years ago, a French entomologist named Pierre Paul Grassé observed [22] that a particular breed of termites are sensible to a chemical substance called *pheromone* that activates a genetically encoded reaction. He discovered that ants deposit this substance along the trail leading to a food source. Other ants, able to smell this substance, can therefore follow the same path and reach the food.

In the famous "binary bridge" experiment [14], some researchers showed experimentally how the pheromone impacts on ant choices. To reach the food, a colony of ants could choose between two bridges of equal length. In the early stage of the experiment, ants chose randomly one of the bridges to reach the food and, on the way back, they deposited pheromone on the path. Due to stochastic oscillations, one of the two bridges attracted more ants than the other and hence more pheromone was left. This, in turn, attracted even more ants and after a few rounds all the ants started to use the same bridge.

These observations have encouraged active research among computer scientists to see if similar techniques could be applied to solve optimization problems. The first successful attempt tackled the Traveling Salesman Problem (TSP) [37], a well-know $\mathcal{NP}$-hard problem. In TSP a set of towns is given and the distance between each of them is known. The goal is to find the shortest route that visits all the towns once and only once. The basic ant protocol works in an iterative fashion. Initially, a number of artificial ants are located at each town and they start exploring the solution space by moving from one town to another. At each step, an ant chooses the next town to visit according to the probability associated with the link that connects its current town to the next one. This probability depends on the pheromone previously accumulated on that link and, optionally, on some heuristics (e.g., the distance between the two towns). After a tour has been completed, on the basis of the quality of the solutions achieved, the pheromone values are modified to influence ants in future iterations. Ant protocols have been shown to perform quite well on the TSP [45].

Dorigo and Di Caro [17] have formalized this approach into a *metaheuristic* for combinatorial problems called *Ant Colony Optimization* (ACO) to provide a conceptual framework for all the ant-based protocols. These protocols have been applied in many different contexts ranging from typical optimization problems like scheduling [13] and set covering [27], to Internet traffic [10, 46] and vehicle routing [9] problems.

The general scheme is provided in Figure 4. At each iteration, a number of solutions are constructed by the ants that may be improved using standard *local search* techniques, and finally, based on the quality of these solutions, the pheromone values are updated. A generic combinatorial problem can be expressed as a graph $G(V, E)$ where the set V represents the components of the solutions (e.g., the towns in the TSP). An ant builds its solution incrementally by moving from a vertex to another along an edge $e \in E$.

1: **loop**
2: Construct Ant Solutions
3: Apply Local Search (optional)
4: Update Pheromones
5: **end loop**

Figure 4: The Ant Colony Optimization Metaheuristic.

The probability of selecting an edge from i to j for a generic ant k is usually given by

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in \mathcal{N}_i^k} (\tau_{il})^\alpha \cdot (\eta_{il})^\beta} \qquad \text{if} j \in \mathcal{N}_i^k$$

where η_{ij} is a priori available heuristic information, α and β are two exponents to weight the influence of the pheromone τ and the heuristic η and $\mathcal{N}_i^k$ is the set of possible choices, e.g., the town not yet visited in the TSP.

Although ant protocols are mostly run in a centralized fashion (as in the TSP), in network routing [15, 46] a fully decentralized version of ant protocols is applied. In these protocols, ants are implemented by messages exploring the network to discover shortest routes and pheromone is used to influence routing decisions. Here, the similarities with the gossip protocols are evident. Both families of protocols rely only on local information and operate iteratively. In addition, ant- and gossip-based protocols are intrinsically probabilistic in nature. Pheromone biases ant choices to converge towards optimal solutions but probability is the key to make the protocol explore alternative, possibly better solutions, thus escaping from local minima. Although traditional gossip protocols assume a uniform random probability, some recent articles [31, 49] propose a similar biased approach for gossip-based overlay management. In these protocols, nodes gossip more frequently with their "good" neighbors to converge towards the desired overlay but some gossip exchanges are made with random nodes to discover potentially better neighbors. Finally, ant and gossip protocols are well-suited to address dynamism, since both are able to quickly adapt to ever-changing conditions and to reconfigure properly.

These common features demonstrate that a strong connection exists between the two families of protocols, which can be exploited in several ways. On the one hand, gossip protocols may benefit from theoretical work developed in the field of ACO (e.g., [25]) to improve the understanding of gossip phenomena. Also, since in many cases the performance of ant protocols is significantly increased when combined with traditional local search techniques, it could be investigated whether this holds for gossip protocols as well. On the other hand, this may represent a case for exporting gossip protocol knowledge to other domains. Indeed, since ant protocols are good at solving combinatorial problems, our conjecture is that gossip protocols might represent a viable tool for addressing these problems too (see [8] for a preliminary study on the topic).

3.3 Self-stabilizing Systems

Self-stabilizing systems have been defined by Dijkstra in [16] as distributed systems satisfying a desired predicate p after a finite number of steps, regardless of the initial state. The system states that satisfy p are called the *legitimate configurations*. The characteristic properties of a self-stabilizing

system include:

1. **Closure:** Once p is established in $\mathcal{S}$, it can not be violated by any legal transition.
2. **Convergence:** Starting from an arbitrary state, $\mathcal{S}$ reaches a state satisfying p within a finite number of transitions.

Years after the work of Dijkstra, considered by Lamport "to be a milestone in work of fault tolerance" [36], self-stabilization came to be widely regarded as a unified approach for tolerance to transient failures.

Interestingly, many aspects of self-stabilizing protocols are shared by gossip protocols. Dijkstra's notion of *privilege* refers to a certain satisfied local condition at a participant. A privilege enables a corresponding transition. Locality is clearly a requirement shared by gossip protocols as well. Furthermore, a privilege must be present in each legitimate configuration ensuring that at any time at least one participant is allowed to make a transition. In gossip protocols, participants act periodically also leading to an infinite execution. Finally, a legitimate configuration must be reached by a self-stabilizing system. Gossip protocols are also designed to achieve a certain global state.

For instance, Dijkstra's seminal protocol [16] introduces a set of $n+1$ nodes arranged as a ring with clockwise increasing ids in $[0, n]$. Let node i have state $x_i \in \{0, \ldots, k-1\}$. Each node continuously evaluates one condition to test for the presence of a privilege. This condition and the corresponding action are different for node 0 and the rest of the nodes:

$$\begin{aligned} 0 < i \leq n: \quad & x_{i-1} \neq x_i \quad \Rightarrow \quad x_i \leftarrow x_{i-1} \\ i = 0: \quad & x_n = x_0 \quad \Rightarrow \quad x_0 \leftarrow (x_0 + 1) \mod k. \end{aligned}$$

The legitimate configurations are configurations where exactly one privilege is present in the network. This protocol self-stabilizes if $k > n$.

The protocol presented above is similar to a gossip protocol. The information stored locally is bounded and small, the number of outgoing neighbors (connected through outgoing links) is a constant (2), hence the protocol is scalable. Moreover, as mentioned above, each node is assumed to act infinitely often while the periodic execution of a node in gossip protocols implies that each node acts infinitely often too.

These observations raise the important question of whether gossip protocols and self-stabilizing protocols are identical. If so, it would be pointless to investigate gossip protocols any further.

On the one hand, it is clear that not all self-stabilizing protocols are gossip-related. For instance, the example described above associates one node (node 0) with a behavior different from other node behaviors. This asymmetry violates one of our prototypical gossip properties.

On the other hand, some gossip protocols do not self-stabilize. There is an important notion of fix-point in self-stabilizing protocols that gossip protocols might lack. We will now give an example of a gossip protocol that has no fix-point. Similar to a self-stabilizing protocol, the ranking protocol [19] solves a global problem. This global problem—known as the *distributed slicing* problem—is for every node to guess which portion (namely the slice) of the set of all values its own value belongs to. The protocol works roughly as follows: each node periodically shuffles its set of neighbors then it sends its value to some neighbors. Each node counts the number g of the last received values and the ℓ values among them that are lower than its own. Based on the uniform drawing of neighbors, this leads to an approximation $\frac{l}{g}$ of its value position in the system. This approximation determines which portion/slice its value seems to belong to.

Even though the ranking protocol makes each node act locally, to periodically gossip with neighbors and to manage a constant amount of information, this protocol does not self-stabilize.

The ranking protocol ensures that after an infinite execution the system reaches a legitimate configuration with a probability 1. That is, it satisfies the convergence property of probabilistic self-stabilization as defined in [43]. However, the closure property of probabilistic self-stabilization is not ensured since the system can arbitrarily reach a legitimate configuration at any time before leaving it. This violation is due to the randomized aspects of this protocol that might join and leave a legitimate configuration in a finite prefix of any execution. Although the ranking protocol like other gossip protocols does not self-stabilize, it solves the distributed slicing task even in the presence of continuous dynamics. That is, it is noteworthy that gossip protocols achieve a desirable result while not necessarily self-stabilizing.

Self-stabilization is clearly a "formal and unified approach to fault tolerance under a model of transient failures" as explained by Schneider [43]. Nowadays, systems are more complex and experience transient failures in a continuous manner due to their scale. Apparently, gossip protocols represent a de facto solution to face these continuous transient failures that self-stabilization can not handle. Two interesting features of self-stabilization are its formalism and its unification. As far as we know, prototypical gossip lacks these two features. A key challenge is to eventually incorporate them into a future gossip definition.

3.4 Cellular Automata

A cellular automaton (CA) [21] is a dynamical system model that is discrete in both space and time. Its original formulation was given by J. von Neumann and S. Ulam in the 1940s. The original idea behind it is to study the essence of the reproduction process, but in a purely abstracted manner. That is why it focuses on the general problem of reproducing information.

A CA consists of an infinite, regular and n-dimensional grid of *cells*. Each cell has a finite set of *states*. The *neighborhood* set of a cell A can be defined as the cells within a radius of range r ($r = 1$ if not otherwise stated). The neighborhood does not change over time. A cell state at time t is a function of the states of its neighborhood at time $t-1$. All cells in the system share a set of state-transition rules based on neighborhood states. Essentially, these rules define a deterministic finite state automaton having a finite set of states Q, a finite set of inputs X and a transition function $\delta : Q \times X \rightarrow Q$. When all the cells in the grid have applied the rule set, a new *generation* in created.

For practical reasons, a CA is simulated on a finite grid, but this can lead to problems when computing the current state of the cells on the border. In two dimensions, for example, we can solve the problem by bending and taping the grid borders in order to obtain a toroidal mesh.

A popular example of a CA is given by J. Conway's "game

of life" [39]. In addition, it is a CA that supports universal computation. The name comes from the binary state allowed for cells: empty (dead) or populated (alive). It is a two dimensional automaton in which cells obey the following simple rules:

1. **loneliness rule**: if a populated cell has less than two neighbors, than it dies (becomes empty)
2. **overcrowding rule**: if a populated cell has more than three neighbors then it dies
3. **reproduction rule**: if an empty cell has exactly three neighbors then it becomes populated
4. **stasis rule**: if a cell has exactly two neighbors, then it does not change its state

Notice that the rules are inspired by the basic properties of an ecosystem where the population density plays a crucial role for reproduction.

The game of life can be started from a random initialization; after successive generations, some stable common patterns will emerge. The patterns can be static or dynamic. They can even replicate (*self-replication*). Using such patterns one can implement basic memory, counting and iteration. In fact, the game of life can emulate a Turing machine.

This brief description of the CA model is far from being exhaustive, but it is sufficient to understand the relationship between CA and gossip. Let us mention some differences first. Instead of talking to a random neighbor in each round, a cell broadcasts a query to every cell in its neighborhood and then needs to collect the replies from all its neighbors in order to compute the next state. Although this broadcast communication model applies to wireless ad-hoc networks, it is not typical in gossip-based protocols. Besides, the basic CA model assumes *strong synchronization* between two distinct generations in order maintain state consistency. This requirement does not exist in prototypical gossip.

The synchrony requirement is somewhat surprising as the CA model is supposed to describe the self-* properties of natural systems in which there is no notion of the existence of a global clock. This motivated the formulation of an *asynchronous cellular automata* (ACA) approach [41]. All the important results achieved in the past 50 years (like self-reproduction and universal computation) hold in the asynchronous model as well.

A theorem guarantees that for each cellular automaton $\mathcal{A}$ on a graph $G(V, E)$, we can build a second cellular automaton $\mathcal{A}'$ on the same graph. For each node $v \in V$, the automaton $\mathcal{A}'$ will have $3n^2$ states, where n is the number of states in automaton $\mathcal{A}$. The construction details of the asynchronous automaton $\mathcal{A}'$ can be found in [41]. The general idea behind this process is that the asynchronous transition rule δ' can distinguish between a "present", "future" and "past" state for neighboring nodes; the transition rule allows a state change in a node v if no node in its neighborhood gets more than one step behind should update be needed; otherwise, there is no change at all. Essentially, the approach ensures some sort of synchronization over time at the neighborhood level and lazy synchronization emerges at a global level.

As opposed to the asynchronous numerical iteration method mentioned in Section 3.1, an ACA has a considerably larger state space than the original, synchronous CA it intends to model, and it has a relatively complicated construction. What happens if we simply run a CA applying asynchronous updates? As reported in [7], this can result in non-trivial behavior for Conway's "game of life". If we update cells probabilistically, instead of doing it in lock-step, we get a wide array of different behaviors depending on the parameters of the update model.

Let us now summarize the similarities with prototypical gossip. As we have seen, synchrony can be relaxed. In addition, in a CA, participants (cells) rely on local knowledge and local interaction. This means that every action is a function of the local states of the participants and the knowledge of the system is local as well, as it is limited by the neighborhood set.

In a CA, the communication scheme is also periodic. The required processing capacity of a single cell is low, and the amount of information needed per message is low as well.

As in gossip, every cell runs the same protocol (e.g., it behaves according to the same set of state-transition rules). Of course, we consider that all cells are benign. Malicious cells may lead to arbitrary behavior.

These obvious similarities can form the basis for an investigation into gossip protocols using the tools typical in CA theory, like the classification of the dynamics into convergent, periodic, and chaotic behavior, and, indirectly, drawing analogies between systems that are typically modeled by CA (including several biological, economic and sociological phenomena), and gossip.

4. EXPORTING IDEAS

Now we will take a completely different view and look at well-known problems in different disciplines, and investigate whether we can export some of the gossip principles to get some of the "good properties" that are normally associated with gossip protocols. The resulting discussion is not as clear as that in the previous section: it could be interpreted as a research agenda or a plan for future work. Only a few preliminary results are available. Nevertheless, all the ideas described here look promising and deserve further investigation.

Before actually presenting the examples, it is important to introduce the rationale we adopted for selecting each one. The idea is to look for research topics where we can apply one of the following principles:

- **A global view is undesirable**: the idea here is to attack problems where existing solutions are (i) either centralized or (ii) are decentralized but strongly depend on each node having a complete (global) view of the input data. The question we have to pose ourselves is the following: is this global view really needed? Can we obtain "reasonably good" results with partial information only? The cooperative protocols of Section 4.1 and the clock synchronization example of Section 4.2 fall into this category.
- **No global view is possible at all**: here the idea is to tackle problems where it is immediately clear that no single node can ever have an up-to-date global view—for example due to the extremely large scale, dynamism and locality of information. Here the idea is to discuss whether a gossip approach could produce benefits in term of robustness and scalability with respect to existing solutions. Examples of this approach

include the routing protocols of Section 4.3, especially those related to MANETS, and the vehicle routing problem of Section 4.4.

4.1 Cooperative Protocols

The recent growth of peer-to-peer and wireless technologies has directed the attention of the scientific community towards models and tools developed in the social sciences to understand and improve network protocols. An example of this emergent trend is the famous file-sharing protocol BitTorrent [11] which is based on the "tit-for-tat" strategy, studied in game theory, to discourage free-riders and improve download speeds. Similar approaches that exploit notions from game theory are being investigated in the wireless domain as well [18], especially in infrastructure-less networks, where users need to cooperate to provide a multi-hop routing service.

Unfortunately, traditional approaches in game theory often assume a complete knowledge of the actors in the system and observable actions: node behavior should always be globally visible. In peer-to-peer or wireless scenarios these assumptions might not hold. The scale of the system prevents complete knowledge and some types of behavior can be influenced by external factors too, like packet loss that could be misinterpreted as selfish action.

We believe that gossip protocols are an interesting paradigm to explore in order to tackle the problem of cooperation. To justify our claim, we will discuss SLACER [28], one of the first contributions that exploits gossip protocols to create self-organized cooperative networks of peers.

Generally speaking, the SLACER algorithm works as follows. Each node in the network is connected to a small set of other nodes in the network which represent its neighbors. Each node is also characterized by a utility function U which indicates how good its performance is (e.g., its score in a game or its download rate in a file-sharing application). This utility is constantly updated at regular intervals. Different nodes may run different strategies, e.g., selfish or cooperative, and hence in general they will have different values of U. Periodically, a node i selects a random node j from the system and compares the two utilities U_i and U_j. If $U_i > U_j$, no action takes place. Otherwise, if $U_i < U_j$, i drops all its neighbors and connects with j's neighbors. In addition, it also adopts the same strategy used by j, e.g., by downloading the code or tuning some parameters, and resetting its utility to zero. After each round, with a low probability, a node may experience a mutation, i.e., it may select a random strategy or replace one of its neighbors with another node randomly chosen from the network.

Simulation results indicate that this protocol converges to 98% of the nodes cooperating, when applied to a typical game like the Prisoner's Dilemma [2], even though at the beginning all the nodes implement a non-cooperative strategy.

These results confirm the suitability of gossip protocols to create and maintain networks of peers with many desirable properties of social networks. This shows that cooperation may represent a fertile area in which gossip protocols can be successfully applied, providing appealing opportunities for researchers to explore.

4.2 Clock Synchronization

Clock synchronization is a fundamental building block for many distributed applications. In a distributed system each host is equipped with a local hardware clock and can communicate with other processes by exchanging messages; clocks embedded on nodes have different drifting rates (i.e. they "tick" with slightly different frequencies), hence their values, even if they are started *exactly* at the same instant, do not remain synchronized. In order to provide applications with a *virtual global clock*, hosts run an algorithm that tries to synchronize local clocks accounting errors due to clock drift and message delivery delay. This topic has been widely studied for almost 20 years, and several algorithms exist which address the problem of clock synchronization in systems with different scales, deployed on both local and wide area networks, with and without external reliable clock sources.

Some of these algorithms [29, 44] employ a *flooding-based* approach where synchronization signals are periodically broadcasted in the system. Each host that receives such a message by another sibling uses it to update its local clock value, taking into account errors introduced by communication channel delays. The periodic update of clock values, the continuous exchange of information among hosts, the use of symmetric algorithms (where each host performs the same actions) characteristic of these solutions make them quite close to our prototypical gossip protocol.

Another widely adopted technique for clock synchronization requires each host to periodically read clock values from neighboring hosts, and to consequently adjust its local value based on previous readings. Synchronization can be achieved against an external reliable time source (as in NTP [40]) or among the independent values maintained by the hosts by imitating the phenomenon of spontaneous synchronization among coupled oscillators. These algorithms could probably be greatly helped by the adoption of a gossip-based communication scheme: the exchange of clock values, initially limited to neighboring hosts, or realized through a structured (usually hierarchical) architecture, would be improved and simplified by the adoption of a peer-sampling service. One recent approach is [3], where a synchronization primitive based on biological models of synchronized firefly flashing was proposed.

Another study in this direction was presented in [30]. This study assumed the presence of a host perfectly synchronized with an external reliable real-time clock source. Each host uses a peer sampling service [32] to select another node in the network and to exchange timing information with it. If the time read from the contacted node is of a higher quality (more synchronized with the external time source) than its own time (e.g. the contacted node is the source), then the reading node will adopt the clock setting of the other one. The quality of timing information is evaluated using a dispersion metric like the one provided by NTP. The adoption of a peer sampling service and a gossip-based interaction makes the algorithm well suited for large-scale dynamic settings, like those characterizing peer-to-peer applications, where more rigid schemes, like those imposed by NTP, reveal their limitations.

Currently, some of the authors of this position paper are studying whether a similar approach could achieve *internal* clock synchronization for very large scale dynamic systems [5]. In this specific case, one of the main problems is how synchronized clocks can be made resilient to "perturbations" stemming from the addition of new non-synchronized clocks

to the system.

4.3 Routing protocols for multi-hop networks

Message routing in multi-hop networks is one of the oldest and most studied problems in the area of computer science. A multi-hop network is simply a network of nodes where each node can communicate only with a subset of all its siblings. A multi-hop network is usually represented through a graph $G(N, E)$ where N is the set of communicating nodes, and E is the set of links interconnecting them. If the graph G is connected, each pair of nodes can communicate by sending messages via a path that links them through a set of network links and nodes. To implement this form of multi-hop communication in a real network there is a need for a *routing protocol* that is able to construct, for each pair of nodes, the path interconnecting them.

RIP [23,24] is a distance-vector routing protocol based on the Bellman-Ford algorithm. Despite the advent of some other protocols like OSPF, RIP is still widely deployed as an interior gateway protocol (IGP) mainly because of its ease of configuration. Distance-vector algorithms build a certain level of knowledge of the system (every other node of the system is eventually known) with local interactions only. The aim of such algorithms is to choose the best route for packets, based on various metrics. RIP's metric is the *hop*, representing a jump from a router to one of its directly connected neighbor routers. Each router hosts and manages a distance vector, containing its distance to each other router of the topology, recorded with the first router on the route to that network.

Let $D(i, j)$ represent the number of hops on the *shortest path* from i to j. The protocol starts with local vectors initialized as follows: 0 for the current router ($D(i, i)$), 1 for its directly connected routers, and infinity for its not yet known routers. Every 30 seconds, each router sends its entire vector table to its neighbors (1-hop neighbors). When receiving a vector from a neighbor k, a router i updates its own vector for each destination router j as $D(i, j) = min([1+D(k, j)], c)$ where c is the old value. This algorithm eventually converges, giving one of the shortest paths for all network destinations to each router.

To improve convergence time, triggered updates are used in addition to the periodic mechanism: when a gateway changes the metric for a route, it sends an update message almost immediately to its neighbors. Only the routers which have their path using the advertising router may change their metric and then propagate the update, and so on.

By looking at this protocol overview, some obvious similarities with our definition of prototypical gossip arise. RIP fits the round-based condition, as the information spread is periodic, based on a default parameter setting. All the participants—here, the routers—also participate with equal responsibilities. Locality does not hold, since the full set of nodes needs to be known (which is of course suitable for small scale networks). However, only network address, distance and next-hop router are kept for each destination site, while OSPF builds a full map of the network on each site, including routers, links and costs between them.

On the other hand, the two remaining parts of the definition are those for which some improvements could be expected through some kind of technology transfer. First, there is no random selection as each neighbor router is systematically contacted to push updates. Secondly, there is no real limited transmission as the vector is either fully pushed every period, or updates are propagated in an epidemic non round-based fashion. As two of the main criticisms about the RIP protocol are the overhead produced by this way of updating information, and the relatively important convergence time, one could imagine using traditional gossip information propagation to improve overall performance.

One approach could be to randomize both neighbor and vector entry selection, pushing just a small subset of available information, and at the same time applying a shorter period: if smaller messages are sent to fewer routers but more often, it is likely that the overhead will not increase. It would thus be interesting to examine the variation of overhead/convergence tradeoff.

While routing protocols for fixed networks (like RIP or OSPF) have remained "anchored" to more classical interaction schemes for historical reasons, the gossip approach has found its way into the new area of information routing for mobile ad-hoc networks (MANET). A MANET is composed of a set of independent nodes that communicate via wireless links. Direct communication between two nodes is realized without the help of any fixed infrastructure, so it is feasible only as long as the two nodes remain within communication range. Nodes belonging to a MANET are supposed to be located within a predefined area where they are free to move. The absence of a fixed communication infrastructure makes communication between two distant nodes possible only through the adoption of a routing protocol that is able to route information though the "spontaneous" multi-hop network emerging from the union of all nodes together with their wireless links. Node movements, however, make the topology of this network evolve continuously, thus discouraging the adoption of classic routing protocols that assume a rather stable interconnection topology. The interesting problems proposed by this peculiar setting led to the proposal of many different solutions. Some of these solutions inherit many of the characteristics of our prototypical gossip protocol.

For example, [26] proposes a simple approach where a deterministic route discovery phase is substituted by a gossip-like query diffusion mechanism. The intrinsic tolerance of approaches based on gossiping to system reconfigurations and node/link failures makes the algorithm better suited to mobile settings than classic routing algorithms that heavily rely on deterministic data structures containing routing information.

Gossip-like techniques can also be recognized in [6]. The proposed algorithm is based on a periodic exchange of beacon signals among nodes; these signals are used to maintain locally at each node routing tables containing probabilistic information about current node positioning; the accuracy of the information a node maintains about another node position is perfect when they are in direct contact, but tends to "degrade" with time as soon as they move out of communication range. The routing phase is realized exploiting this information and probabilistically forwarding a packet through nodes that maintain more accurate information about the destination position, and that are, therefore, closer to it with a higher probability. This algorithm respects many of the properties of our prototypical gossip algorithm: (i) the same algorithm is executed by each node, (ii) it is in some sense round-based as beacon signals must be sent by every node with the same period to ensure that a meaningful connection

holds between out-of-date information maintained in routing tables and the relative position of nodes, (iii) the amount of information periodically sent by each node is limited to the beacon signal and finally (iv) only local information is maintained and used during the routing phase.

4.4 Vehicle Traffic Control

An important problem that could be efficiently attacked through gossip protocols is the *autonomous vehicle control* problem [47], where a large collection of independent vehicles must travel from their source to their destination as fast as possible, interacting with their peers to avoid collisions.

Currently, vehicle control is left to humans; interestingly enough, they already adopt a degenerate form of "gossip-based" communication. Consider, for example, the formation of queues on highways. Here, communication is performed through brake lights: a single vehicle accidently brakes; starting from it, each vehicle communicates with the following one; a "brake wave" propagates along the highway, until the traffic stops completely. When finally vehicles are able to start again, nothing in front of them can explain the formation of the queue.

We understand how futuristic a "driver-less" traffic control system must sound, so we will limit ourselves to propose a simpler problem that can be solved appropriately via a gossip approach and existing technology. The idea is to provide human drivers with congestion information about the current situation in the path between their sources and destinations. Clearly, a global view of the system is out of the question for it cannot be centralized or replicated among vehicles. Locality plays a key role: We are not interested in the current traffic state hundreds of kilometers away, or nearby, recently left areas.

An increasing number of modern cars are already equipped with navigators that contain a collection of useful "sensors" about current position, direction, and speed. This information, coupled with a precise knowledge of the area, is exactly what is required to evaluate traffic conditions. Car-to-car communication networks are being discussed both in academic [20] and industrial research [1]. Gossip may be the glue to enable these two technologies to act together. We can foresee a system where information about traffic conditions is aggregated [33] and broadcasted using gossip; this information can be provided as input for the navigator that could, for example, suggest alternative routes in the case of congestion.

5. CONCLUSIONS

In this position paper we laid emphasis on the importance of interdisciplinary explorations, and this should have two main benefits. First, we can import techniques and tools to tackle problems in the field of gossip-based protocols, and we can also export gossip technology by re-interpreting some problems and systems as gossip-based, which should allow the application of gossip algorithms and theoretical results in those fields. Second, interdisciplinary comparison not only focuses on similarities but also on differences, and this should help us crystallize the key properties of the class of gossip-based algorithms, and in time lead to a more formal definition.

Our hope is that this paper will, directly or indirectly, inspire new directions of research by broadening the perspective on gossip protocols.

6. REFERENCES

[1] BMW Connected Drive, July 2004. `http://www.connected-drive.de`.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic books, New York, US, 1984.

[3] O. Babaoglu, T. Binci, M. Jelasity, and A. Montresor. Firefly-inspired heartbeat synchronization in overlay networks. In *First IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007)*, 2007.

[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide*. SIAM, Philadelphia, 2000.

[5] R. Baldoni, A. Corsaro, L. Querzoni, S. Scipioni, and S. Tucci-Piergiovanni. An adaptive coupling-based algorithm for internal clock synchronization of large scale dynamic systems. Technical report, MidLab 2/07 - Università degli Studi di Roma "La Sapienza", 2007.

[6] R. Beraldi, L. Querzoni, and R. Baldoni. A hint-based probabilistic protocol for unicast communications in manets. *Ad Hoc Networks*, 4(5):547–566, 2006.

[7] H. J. Blok and B. Bergersen. Synchronous versus asynchronous updating in the "game of life". *Phys. Rev. E*, 59:3876–9, 1999. `http://rikblok.shorturl.com/lib/blok99.html`.

[8] M. Brunato, R. Battiti, and A. Montresor. GOSH! Gossiping Optimization Search Heuristics. In *Proceedings of the Learning and Intelligent Optimization Workshop (LION 2007)*, Andalo, Italy, 2007.

[9] B. Bullnheimer, R. Hartl, and C. Strauss. An Improved Ant System Algorithm for the Vehicle Routing Problem. *Annals of Operations Research*, 89:319–328, 1999.

[10] G. D. Caro and M. Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.

[11] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proceedings of the* 1^{st} *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, 2003.

[12] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.

[13] M. den Besten, T. Stützle, and M. Dorigo. Ant colony optimization for the total weighted tardiness problem. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 611–620, London, UK, 2000. Springer-Verlag.

[14] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[15] G. Di Caro, F. Ducatelle, and L. Gambardella. AntHocNet: An Adaptive Nature-inspired Algorithm for Routing in Mobile Ad Hoc Networks. *European Transactions on Telecommunications*, 15(4), 2005.

[16] E. W. Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[17] M. Dorigo and G. D. Caro. *New Ideas in Optimization*, chapter The Ant Colony Optimization metaheuristic, pages 11–32. McGraw Hill, London, UK, 1999.

[18] M. Felegyhazi and J. P. Hubaux. Game theory in wireless networks: A tutorial. Technical report, EPFL, 2006.

[19] A. Fernández, V. Gramoli, E. Jiménez, A.-M. Kermarrec, and M. Raynal. Distributed slicing in dynamic systems. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07)*. IEEE Computer Society, 2007.

[20] A. Festag, H. Fußler, H. Hartenstein, A. Sarma, and R. Schmitz. FLEETNET: Bringing car-to-car communication into the real world. *Computer*, 4(L15):16.

[21] G. W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. The MIT Press, 2000.

[22] P. P. Grassé. *Les Insectes Dans Leur Univers*. Ed. du Palais de la decouverte, Paris, France, 1946.

[23] N. W. Group. Routing information protocol. *rfc 1058*, 1988.

[24] N. W. Group. Rip version 2. *rfc 2453*, 1998.

[25] W. J. Gutjahr. On the finite-time dynamics of ant colony optimization. *Methodology And Computing In Applied Probability*, 8(1):105–133, 2006.

[26] Z. Haas and J. H. L. Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, 2006.

[27] R. Hadji, M. Rahoual, E. Talbi, and V. Bachelet. Ant colonies for the set covering problem. In *In Proceedings of ANTS2000: From Ant Colonies to Artificial Ants*, pages 63–66, Bruxelles, 2000.

[28] D. Hales and S. Arteconi. Slacer: A self-organizing protocol for coordination in peer-to-peer networks. *IEEE Intelligent Systems*, 21(2):29–35, 2006.

[29] J. Y. Halpern, B. Simons, R. Strong, and D. Dolev. Fault-tolerant clock synchronization. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 89–102, New York, NY, USA, 1984. ACM Press.

[30] K. Iwanicki, M. van Steen, and S. Voulgaris. Gossip-based clock synchronization for large decentralized systems. In *Proceedings of the Second IEEE International Workshop on Self-Managed Networks, Systems and Services (SelfMan 2006)*, pages 28–42, Dublin, Ireland, June 2006.

[31] M. Jelasity and O. Babaoglu. T-Man: Gossip-based overlay topology management. In S. A. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, editors, *Engineering Self-Organising Systems: Third International Workshop (ESOA 2005), Revised Selected Papers*, volume 3910 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2006.

[32] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[33] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.

[34] H. Johansen, A. Allavena, and R. van Renesse. Fireflies: Scalable support for intrusion-tolerant network overlays. In *Proc. EuroSys 2006*, 2006.

[35] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491. IEEE Computer Society, 2003.

[36] L. Lamport. Solved problems, unsolved problems and nonproblems in concurrency. In *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1984.

[37] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Travelling Salesman Problem*. John Wiley & Sons, Chichester, UK, 1985.

[38] B. Lubachevsky and D. Mitra. A chaotic asynchronous algorithm for computing the fixed point of a nonnegative matrix of unit radius. *Journal of the ACM*, 33(1):130–150, January 1986.

[39] J. Matthews. Conway's game of life project. 5 2000.

[40] D. L. Mills. Network time protocol (version 1) specification and implementation.

[41] C. L. Nehaniv. Self-reproduction in asynchronous cellular automata. *eh*, 00:201, 2002.

[42] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[43] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, 1993.

[44] T. K. Srikanth and S. Toueg. Optimal clock synchronization. *J. ACM*, 34(3):626–645, 1987.

[45] T. Stützle and M. Dorigo. *Evolutionary Algorithms in Engineering and Computer Science*, chapter ACO algorithms for the traveling salesman problem, pages 163–183. John Wiley & Sons, Chichester, UK, 1999.

[46] D. Subramanian, P. Druschel, and J. Chen. Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks. In *Proc. Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 832–839, Nagoya, Japan, 1997.

[47] R. Sumner. In-vehicle traffic congestion information system, Nov. 17 1992. US Patent 5,164,904.

[48] R. van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.

[49] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In J. C. Cunha and P. D. Medeiros, editors, *Proc. Euro-Par*, number 3648 in Lecture Notes in Computer Science, pages 1143–1152. Springer, 2005.