PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2008) **366**, 3747–3757
doi:10.1098/rsta.2008.0122
Published online 31 July 2008

Self-∗ properties through gossiping

By Ozalp Babaoglu[1,*] and Márk Jelasity[2]

[1]*University of Bologna, Mura Anteo Zamboni 7, 40126 Bologna, Italy*
[2]*University of Szeged and HAS, PO Box 652, 6701 Szeged, Hungary*

As computer systems have become more complex, numerous competing approaches have been proposed for these systems to self-configure, self-manage, self-repair, etc. such that human intervention in their operation can be minimized. In ubiquitous systems, this has always been a central issue as well. In this paper, we overview techniques to implement self-∗ properties in large-scale, decentralized networks through bio-inspired techniques in general, and gossip-based algorithms in particular. We believe that gossip-based algorithms could be an important inspiration for solving problems in ubiquitous computing as well. As an example, we outline a novel approach to arrange large numbers of mobile agents (e.g. vehicles, rescue teams carrying mobile devices) into different formations in a totally decentralized manner. The approach is inspired by the biological mechanism of cell sorting via differential adhesion, as well as by our earlier work in self-organizing peer-to-peer overlay networks.

Keywords: self-organization; epidemics; gossip; overlay networks

1. Introduction

Computer systems of today are extremely complex, highly distributed, heterogeneous and dynamic. Almost all computing devices are becoming interconnected through various forms of wired or wireless networks, and have access to—and might also provide—global or local services over the network. Even the distinction between centralized and distributed systems is becoming increasingly blurred: the largest computer centres today may contain hundreds of thousands of unreliable computers, perhaps geographically distributed as well, referred to as cloud computing (Hand 2007).

Making sure that these extremely complex systems work as they are supposed to is challenging. One important challenge is that complex systems often have properties that are not 'designed' into them but instead emerge from their evolution and interactions with their environment. Well-known examples include the complex network structure of the World Wide Web, the physical Internet or computer virus spreading patterns resulting from the interaction of user behaviour and complex networks. Recently, complex social networks have started playing an increasing role through Web 2.0 and peer-to-peer technologies where they shape the content of the services as well as their usage patterns (such as flash crowds and BitTorrent traffic), which adds further complexity.

* Author for correspondence (babaoglu@cs.unibo.it).

One contribution of 19 to a Discussion Meeting Issue 'From computers to ubiquitous computing, by 2020'.

A traditional approach to automatically managing complex systems involves control loops (Kephart & Chess 2003), where the idea is to replace human operators with controllers that monitor the system and perform corrective actions. In the light of the above observations, however, we emphasize the importance of understanding the self-organizing, emergent characteristics of complex systems. This knowledge is essential for building control loops as well, but taking the idea one step further, we believe that we could also influence or even engineer *desirable* emergent behaviours from bottom up.

Engineering emergence through well-designed local interactions to achieve a desired aggregate global behaviour is potentially much cheaper and simpler to implement, as it does not require specialized infrastructures. In particular, in the context of ubiquitous computing, where devices can easily interact locally but where global, unicast-style communication is expensive to implement, this line of thinking is especially promising.

In this paper, we outline our approach to designing global behaviour based on local interaction through the analogy of gossiping in §2. In §3, to provide inspiration and to illustrate that gossiping could be a relevant paradigm in ubiquitous computing, we describe a novel protocol for allowing a set of mobile nodes to self-organize into formations. Section 4 concludes the paper.

2. Gossip protocols

The term 'gossip protocol' was probably used for the first time in the seminal work of Demers *et al.* (1987), where they proposed protocols for spreading updates to replicas of a database. The idea in a nutshell is that all nodes execute the same simple algorithm, which periodically selects a random member from the network, and sends and/or requests fresh updates to/from the selected node. It can be shown that as a result, new updates spread in logarithmic time over the network, much like gossiping in social networks.

Subsequently, the concept has come to be interpreted more widely, and now covers many protocols that share some key characteristics such as periodic communication, information exchange with random neighbours and rapid convergence (Kermarrec & van Steen 2007). Instead of attempting to give a more exact definition, we first overview a few gossip-based implementations for different functions in wired networks, and describe how these functions can be built on top of each other. Later on, we argue for the usefulness of this design philosophy and the particular services in a ubiquitous computing setting.

(*a*) *Gossip-based services and protocols in wired networks*

Algorithm 1 illustrates the skeleton of a typical gossip protocol. The basic idea is that at regular time intervals, each node exchanges information with a peer node (usually selected randomly) from the network, followed by updating its local state based on the information exchange. The generic algorithm is instantiated by specifying the local state and providing implementations for the peer selection mechanism and the state update method. One can also obtain push-only, or pull-only versions by removing the symmetric communication step.

Algorithm 1. The gossip algorithm skeleton.

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

8:  procedure onReceive(m)
9:      p ← m.sender
10:     state_p ← m.state                 ▷ push
11:     send state to p                   ▷ pull
12:     state ← update(state_p)
13: end procedure
```

(i) *Information dissemination*

Perhaps the simplest instance of this skeleton is the propagation of updates among replicas of a database. There are numerous variations of this algorithm: we describe only the anti-entropy gossip version here (Demers *et al.* 1987).

In this case, state of a node is the database replica it hosts and method selectPeer( ) returns a random node from the network. During the exchange, the two communicating nodes resolve the differences between their copies of the database, and subsequently both update their own copy applying the new updates learned from the peer.

The attractiveness of this probabilistic approach to spreading new information lies in its simplicity and robustness to benign failures: message loss or crashing nodes are tolerated extremely well. In addition, performance of the algorithm is also very favourable: most variants of the algorithm will spread any new update in the entire network in $O(\log N)$ time steps (where N is network size) with high probability.

(ii) *Peer sampling*

One key component in the previous example was method selectPeer( ) that returns a random sample from the network. This service, which we call *peer sampling*, is important in a number of other distributed applications as well. In very large dynamic systems, the full list of nodes is typically not available at all nodes, so implementing this service is non-trivial.

Interestingly, the gossip paradigm offers a natural approach for implementing the very service it relies on. In this realization, state of a node consists of a small random sample from the network called a *view*, and method update( ) simply merges the view received from the peer with the local view, and keeps, for example, a random subset of the resulting view. An extensive discussion of this service and its variations is provided by Jelasity *et al.* (2007*b*). Recently, a secure version has also been proposed by Bortnikov *et al.* (2008).

As a service, method selectPeer( ) is implemented by picking a member from the nodes local view, without relying on any external information. This fact puts peer selection in a special position that we will discuss later in connection with the component architecture of gossip protocols.

(iii) *Overlay network construction*

Taking a closer look, gossip-based peer sampling service in fact maintains a random overlay network, since the local views can be interpreted as defining overlay network links. Keeping this observation in mind, it is not difficult to see

that the idea could be generalized to create and maintain not only random but also other specialized overlay topologies.

Indeed, as we have demonstrated previously (Jelasity & Babaoglu 2006), a wide range of network topologies can be evolved with a slight modification of the peer sampling algorithm. The state of each node is still a view, which defines the overlay network. The update method, however, is specially designed: in selecting which links to keep, it keeps those that are 'most preferable' for a given node. The peer selection mechanism is also biased towards preferable nodes.

A large number of specialized overlay protocols (e.g. Voulgaris & van Steen 2005; Patel *et al.* 2006; Bonnet *et al.* 2007) follow similar principles.

(iv) *Data aggregation*

In this application, we are interested in calculating some global function over locally available data. For example, the average of the values of some attribute associated with a node (e.g. free storage, CPU capacity, etc).

This is a large area of research; here we very briefly outline our own gossip-based approach to averaging (Jelasity *et al.* 2005). We define the state of a node to be its current approximation of the true average. Method update( ) takes the local approximation and the approximation received from the peer and sets the average of these as the new state. It can be easily seen that all local approximations will converge to the true global average due to mass conservation (the sum of the approximations is constant) and due to the reduction of variance in each step. We have also shown that the convergence is exponential: variance decreases by a constant multiplicative factor in each round.

The idea can also be generalized: we show that one can calculate the maximum, minimum and various other means such as geometric, harmonic, etc. In addition, using the combination of various means, variance, system size and other more complex aggregates can also be calculated.

(v) *Modularity*

The instances of the gossip algorithm described above are not isolated, but instead can be considered services from which to build more complex applications and services (Babaoglu *et al.* 2005).

For example, consider that all algorithms rely on peer sampling, a key service to any gossip protocol. Peer sampling, a gossip protocol itself, is thus the 'lowest layer' of the architecture. Other components such as aggregation can also serve as a service to more complex applications such as load balancing or distributed eigenvector calculation (Jelasity *et al.* 2007*a*), etc.

(*b*) *Gossip and self-organization for ubiquitous computing*

The services we have described so far were all implemented in the application layer, assuming that lower network layers (typically TCP/IP) implement a routing service. In such an environment, almost any overlay network topology is feasible.

In the area of ubiquitous computing, where ad hoc wireless communication constrained by physical space is the norm, significant effort has been devoted to implementing similar routing services, so as to be able to use abstraction layers similar to those of wired networks. In fact, the grand vision of 'Internet of things'

is often associated with ubiquitous computing, where 'smart' physical objects, equipped with identification, sensors and computing power are integrated into the current Internet architecture such as web search and web services.

Without a strong dedicated infrastructure support, however, routing protocols that implement a unicast abstraction in ad hoc wireless networks are often prohibitively expensive; besides, they are not always very useful either. For example, in vehicular networks—a very important application of ubiquitous computing for safety, communication and navigation—this very observation has been made by the German 'Network on Wheels' industrial consortium (Füßler *et al.* 2007). What they found to be crucial instead were more local communication and diffusion-like mechanisms we have described previously.

We believe that in many cases, it is more desirable to implement functionality relying on simpler services such as peer sampling or aggregation. These functions naturally map onto network environments where they can be implemented through gossiping, making use of the emergent physical communication networks defined by proximity and contact, and without making use of a routing service.

Of course, implementing these basic services requires a deeper understanding of the underlying network structure. Topologies of wireless networks depend on the transmission range and mobility patterns of nodes. They can range from static topologies (e.g. sensor networks) to practically fully connected networks, where the range of the nodes covers the area, or where we adopt certain disorganized, dynamic mobility models (like the infamous random way-point model) where all pairs of nodes will get within range in a relatively short expected time.

Recently, more realistic mobility patterns have been studied which result in communication networks with complex emergent properties. An increasing number of empirical studies are being carried out to model the long-term mobility patterns of humans, for example. The properties of the resulting contact networks will certainly play a key role in designing the self-organizing ubiquitous systems of the future, from the point of view of security and efficiency as well (Birman 2007; Kleinberg 2007).

Another, perhaps even more interesting, direction is not to make use of mobility patterns to implement basic services and applications, but to *influence* the mobility patterns of the nodes so as to achieve particular functions. For example, rescue teams, military units, swarms of robots or satellites often need to organize into formations to carry out their tasks efficiently. This can be achieved by adapting gossip-based ideas and protocols that have proven successful in overlay networks. In §3, we illustrate this idea through an example that was inspired by our earlier work on self-organizing overlay networks (Jelasity & Babaoglu 2006).

3. An example: self-organizing patterns of swarms

In this section, we present a protocol that allows mobile nodes to self-organize into a pre-specified global formation. The protocol is shown to work at large scales, with several thousands of nodes and more. The approach does not require any consensus among the nodes, such as global averages or leader nodes. An arbitrary set of nodes can be removed after which the system self-heals automatically using the remaining nodes to recreate the formation without any special action.

(a) System model

We assume that initially we are given a set of N mobile nodes. These nodes can correspond to mobile robots, vehicles, satellites or human beings (e.g. a rescue team) equipped with appropriate computing devices. The nodes are assumed to be able to control the direction and speed of their motion (directly, or indirectly through their human owners). The nodes are capable of wireless communication within a fixed range. The set of nodes can change: nodes can leave or join at any time.

Any node can determine the relative position of any other node (direction and distance) and can move in any direction. Although it is not strictly required, the simplest way to implement this is by nodes having GPS capabilities, as we will assume in this paper.

All nodes have a unique ID that can be generated randomly or assigned based on the properties of a node. These IDs, along with the relative position information, will be used by the nodes to self-organize into the required formation.

Finally, the nodes have access to a peer sampling service similar to the one described in §2. They use this service to obtain the position and ID of a small set of random nodes periodically. Implementations of the peer sampling service for mobile environments are known (Bar-Yossef *et al.* 2006). Accordingly, our approach does not require that the communication range of the devices covers the entire area. Yet for simplicity, in this paper we assume it does, and we propose an extremely cheap and straightforward (although specialized) implementation of the sampling service based on this assumption.

(b) Algorithm description

All nodes execute algorithm 2, which consists of an infinite loop with a delay of Δ time units in each cycle, which determines the speed of motion.

In line 3, the algorithm invokes the peer sampling service. Method getRandomPeers( ) returns descriptors of k random nodes from the entire population of nodes. A descriptor contains the position and ID of the node. We implement the peer sampling service as follows, based on the assumption that the communication range of a node covers all other nodes. In other words, all nodes can communicate with each other directly. For simplicity, let us also assume that the nodes have a rough approximation of the number of participating nodes N (algorithms for approximating the number of nodes could be given as well). Now, let all nodes broadcast their current descriptor with probability k/N. This way, all nodes will receive k descriptors on average from random nodes. Since k is small, the probability of collision is also small, especially if desynchronization is also applied (Patel *et al.* 2007). In addition, if N is large, the communication cost for one node is very small. With this implementation of peer sampling, all nodes will receive an identical set of random node descriptors. According to our simulation results, this has no negative effect on the performance of the algorithm.

Lines 4–5 select the positions of the nodes that are the most and least preferable as neighbours. This is done by ordering the random set of peers according to preference, and returning the first and last elements according to this order. It is this ordering that implicitly determines the formation that the nodes converge to. Examples of formations and their corresponding orderings will be given later.

Algorithm 2. The algorithm run at all nodes for calculating the motion vector.

```
1: loop
2:     wait(Δ)
3:     S ← getRadomPeers( )
4:     p_min ← getMostPreferred(S)              ▷ Neighbour we like most
5:     p_max ← getLeastPreferred(S)             ▷ Neighbour we like least
6:     d ← p_min − getPosition( )               ▷ Direction of p_min from current position
7:     factor ← (D − ‖getPosition() − p_max‖)/D
8:     d ← d + factor · (getPosition( ) − p_max)   ▷ Away from p_max if too close
9:     ExecuteMove(d)                           ▷ Cuts step length at d_max
10: end loop
```

Lines 6–8 calculate the motion vector d. In line 6, we initialize d with a vector pointing to $p_{\min}$, the most preferable neighbour. In line 8, we modify d by adding a vector typically pointing away from $p_{\max}$, the least preferable neighbour. However, the term corresponding to $p_{\max}$ is modified by a factor that can even be negative, thereby reverting the direction of the added vector. The factor is calculated in line 7 where parameter D determines the diameter of the area that should contain the converged formation. In other words, D controls the size of the final formation. The main idea is that if two nodes are more than D apart, then they should start attracting each other, irrespective of their preference. Repelling force is the largest when the distance is zero (where factor is 1). The linear formula in line 7 satisfies these constraints.

In line 9, the calculated vector d is applied to perform the next step. Movement is performed in the direction and magnitude of d, except if d is larger than $d_{\max}$, a parameter corresponding to the maximum step size. In this case, the largest step size is applied.

(*c*) *Experiments*

We performed experiments using the PeerSim simulator.[1] The common parameters and settings for all the experiments with the algorithm were $N=5000$, $D=1$ and $d_{\max}=0.01$. The IDs 1, 2, …, N were assigned to the N nodes. The initial position of each node was a random coordinate in a unit square.

We illustrate the protocol using three different implementations for the preference ordering, a key component as described above. The first implementation orders the set of k nodes according to the *ring distance* from the local ID. That is, we define a circular ID space in which the IDs are ordered in increasing order, except that the maximal ID is followed by the minimal ID. The k nodes are then ordered according to the increasing minimal distance from the local ID on this ring. The most preferred node in this ordering is the one that is closest on the ring. The target configuration with this ordering is expected to be a ring, if the IDs are evenly distributed in the network. If they are not evenly distributed, the target configuration could consist of many disconnected ring-segments, depending on the distribution of IDs.

[1] http://peersim.sourceforge.net/.

A variation of the previous implementation is the *self-healing ring*, which has a connected ring as target configuration, *irrespective* of the distribution of the IDs. To achieve this effect, we still work with the same circular ID space. However, when ordering the k nodes, we distinguish between left and right neighbours. That is, the first two elements in the ordering will be the first neighbour to the left and right, respectively, and so on. That is, the most preferred neighbour will be the closest neighbour to the right or left, with equal probability, irrespective of their actual distance on the ring.

Finally, to illustrate that the algorithm can support formations other than a ring, we implement a *cross distance*-based ordering. Here, the ID space is arranged in a cross shape, where the first half of the IDs forms the first line in the cross and the second half forms an orthogonal crossing line. We then order the k nodes according to the Euclidian distance from the local ID in this virtual cross-shaped ID space. The target configuration of this setting is a cross-shaped formation, if the IDs are evenly distributed.

Figure 1 shows illustrations of the evolution of the formation of the mobile nodes. Each dot represents a mobile node, and the colour of the dot is proportional to its ID. For the experiments with the ring and cross configurations, we set $k=4$, and for the self-healing ring experiment $k=8$. One cycle represents one iteration of the infinite loop in algorithm 2. Note that since the maximum step size $d_{\max}$ is 0.01, a node needs at least 100 cycles to move across the unit square.

In the case of the self-healing ring, at cycle 300 we removed 80 per cent of the nodes that belonged to the continuous range of IDs $(0,10N/8)$. The remaining small segment of 1000 nodes forms a complete ring of the original size relatively quickly.

4. Related work and conclusions

We mention two classes of related work that adopt similar goals but rather different approaches. Christensen *et al.* (2007) put the emphasis on realistic simulations and even implementation in hardware; consequently, the models are of relatively small scale. Nevertheless, the goal is to build global structures based on strictly local communication. It is assumed that there is a 'seed' node, and the shape is 'grown' around this node.

Another example is the work of Ravichandran *et al.* (2007) where, under assumptions similar to ours, the basic idea is that each node first attempts to find out about its own position within the network, and subsequently calculates its target position. This is accomplished through a hierarchical median estimation algorithm. Our approach is potentially more robust and more scalable as well.

Although our approach was presented in its simplest possible form without a rigorous analysis, it can be seen that the basic idea scales well, it results in relatively rapid convergence and it is potentially flexible regarding target formations.

The basic idea of our approach is inspired by the biological mechanism of cell sorting via differential adhesion (Graner & Glazier 1992), where different cell types attract or repel each other, and a mix of cells self-organizes into different configurations based on the parameters of the system. In biological models, there

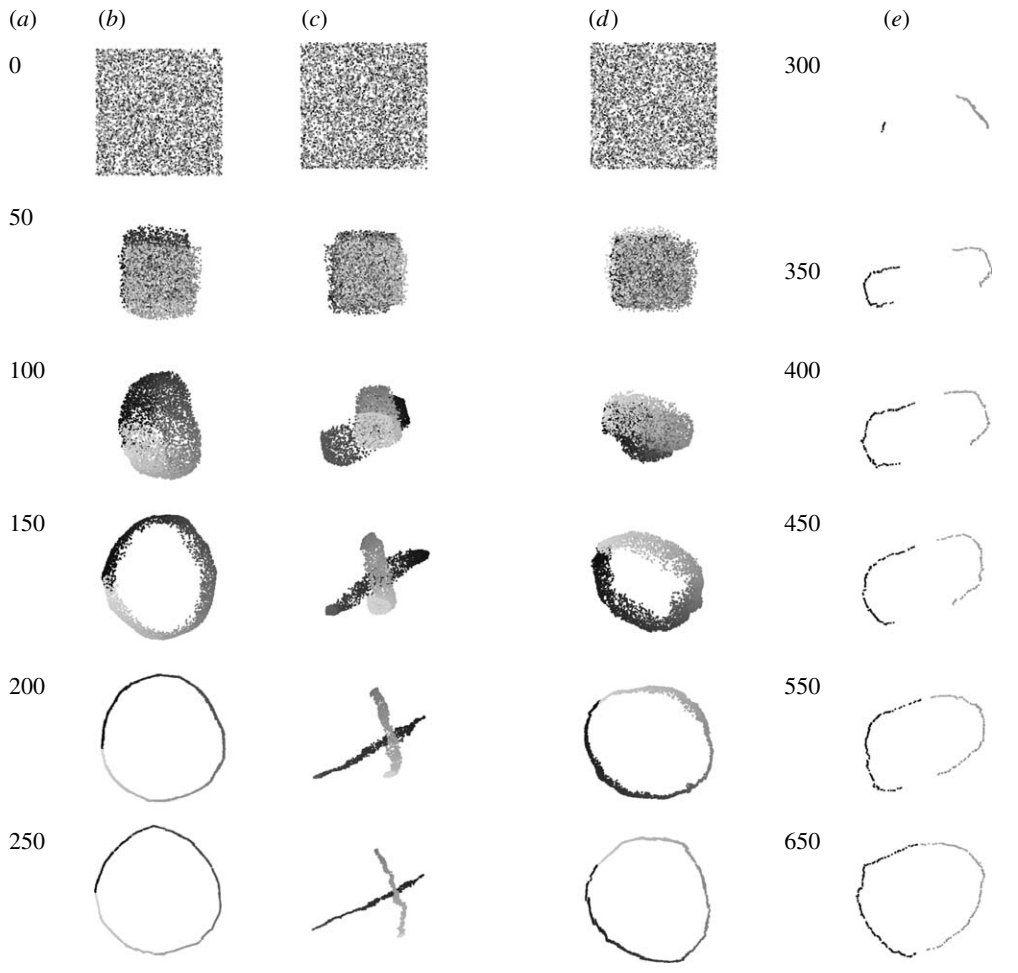

Figure 1. Simulations with 5000 nodes using the (*b*) ring and (*c*) cross target formations until (*a*) cycle 250, and (*d*) self-healing ring with the (*e*) evolution until cycle 650 after damage occurring in cycle 300.

are only a few cell types. Apart from the different model for motion and communication, we generalized this idea and allowed all nodes ('cells') to have a unique ID and unique behaviour based on this ID. Our long-term goal is to be able to engineer local behaviour to create and maintain an arbitrary given formation as we have done in earlier work on self-organizing overlay networks (Jelasity & Babaoglu 2006).

As computer systems become more complex, we argue that it is increasingly important for us not to consider them as passive subjects of external control. We need to incorporate into their operation emergent behaviour that was not intentionally designed. At the same time, we need to continue our efforts towards understanding how to design systems that are 'inherently' self-managing, without external components or agents: in some application areas, this might very well be the only feasible approach to system management.

Partial support for this work was provided by the FET unit of the European Commission through projects BISON (IST-38923) and DELIS (IST-01907). M.J. was supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

References

Babaoglu, O., Jelasity, M. & Montresor, A. 2005 Grassroots approach to self-management in large-scale distributed systems. In *Unconventional programming paradigms* (*UPP 2004*) (eds J.-P. Banâtre, P. Fradet, J.-L. Giavitto & O. Michel). Lecture Notes in Computer Science, vol. 3566, pp. 286–296. Berlin, Germany: Springer.

Bar-Yossef, Z., Friedman, R., & Kliot, G. 2006 RaWMS—random walk based lightweight membership service for wireless ad hoc networks. In *Proc. 7th ACM Int. Symposium on Mobile ad hoc Networking and Computing* (*MobiHoc '06*), pp. 238–249. New York, NY: ACM.

Birman, K. 2007 The promise, and limitations, of gossip protocols. *SIGOPS Oper. Syst. Rev.* **41**, 8–13. (doi:10.1145/1317379.1317382)

Bonnet, F. Kermarrec, A.-M. & Raynal, M. 2007 Small-world networks: from theoretical bounds to practical systems. In *Principles of distributed systems*, vol. 4878, pp. 372–385. Berlin, Germany: Springer.

Bortnikov, E., Gurevich, M., Keidar, I., Kliot, G. & Shraer, A. 2008 Brahms: Byzantine resilient random membership sampling. In *Proc. 27th ACM Symp. on Principles of Distributed Computing* (*PODC'08*).

Christensen, A. L., O'Grady, R. & Dorigo, M. 2007 A mechanism to self-assemble patterns with autonomous robots. In *Advances in artificial life. Proc. 9th European Conference on Artificial Life* (*ECAL 2007*) (ed. F. A. e Costa). Lecture Notes in Artificial Intelligence, vol. 4648, pp. 716–725. Berlin, Germany: Springer.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D. & Terry, D. 1987 Epidemic algorithms for replicated database maintenance. In *Proc. 6th Annual ACM Symposium on Principles of Distributed Computing* (*PODC'87*), pp. 1–12. Vancouver, Canada: ACM Press.

Füßler, H., Schnaufer, S., Transier, M. & Effelsberg, W. 2007 Vehicular *ad-hoc* networks: from vision to reality and back. In *4th Ann. Conf. on Wireless on Demand Network Systems and Services* (*WONS '07*), pp. 80–83.

Graner, F. & Glazier, J. A. 1992 Simulation of biological cell-sorting using a two-dimensional extended potts model. *Phys. Rev. Lett.* **69**, 2013–2016. (doi:10.1103/PhysRevLett.69.2013)

Hand, E. 2007 Head in the clouds. *Nature* **449**, 963. (doi:10.1038/449963a)

Jelasity, M. & Babaoglu, O. 2006 T-Man: gossip-based overlay topology management. In *Engineering Self-Organising Systems: 3rd International Workshop* (*ESOA 2005*), *Revised Selected Papers*, vol. 3910 (eds S. A. Brueckner G. Di Marzo Serugendo, D. Hales & F. Zambonelli). Lecture Notes in Computer Science, pp. 1–15. Berlin, Germany: Springer.

Jelasity, M., Montresor, A. & Babaoglu, O. 2005 Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.* **23**, 219–252. (doi:10.1145/1082469.1082470)

Jelasity, M., Canright, G. & Engø-Monsen, K. 2007*a* Asynchronous distributed power iteration with gossip-based normalization. In *Euro-Par 2007*, vol. 4641 (eds A.-M. Kermarrec L. Bougé & T. Priol). Lecture Notes in Computer Science. Berlin, Germany: Springer.

Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M. & van Steen, M. 2007*b* Gossip-based peer sampling. *ACM Trans. Comput. Syst.* **25**, 8. (doi:10.1145/1275517.1275520)

Kephart, J. O. & Chess, D. M. 2003 The vision of autonomic computing. *IEEE Comput.* **36**, 41–50. (doi:10.1109/MC.2003.1160055)

Kermarrec, A.-M. & van Steen, M. (eds) 2007. *ACM SIGOPS operating systems review 41.* Special issue on gossip-based networking.

Kleinberg, J. 2007 The wireless epidemic. *Nature* **449**, 287–288. (doi:10.1038/449287a)

Patel, A., Degesys, J. & Nagpal, R. 2007 Desynchronization: the theory of self-organizing algorithms for round-robin scheduling. In *IEEE Conf. on Self-Adaptive and Self-Organizing Systems* (*SASO*), pp. 87–96. Los Alamitos, CA: IEEE Computer Society.

Patel, J. A., Gupta, I. & Contractor, N. 2006 JetStream: achieving predictable gossip dissemination by leveraging social network principles. In *Proc. 5th IEEE Int. Symp. on Network Computing and Applications* (*NCA 2006*), *Cambridge, MA,* pp. 32–39.

Ravichandran, R., Gordon, G., & Goldstein, S. C. 2007 A scalable distributed algorithm for shape transformation in multi-robot systems. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2007* (*IROS 2007*), *San Diego, CA,* pp. 4188–4193.

Voulgaris, S. & van Steen, M. 2005. Epidemic-style management of semantic overlays for content-based searching. In *Proc. of Euro-Par* (eds J. C. Cunha & P. D. Medeiros). Lecture Notes in Computer Science, no. 3648, pp. 1143–1152. Berlin, Germany: Springer.