

The background of the page features a large, faint, golden seal of the University of Bologna. The seal is circular and contains a central shield with a cross, surrounded by various figures and architectural elements. The Latin text 'UNIVERSITAS BOLOGNENSIS' is visible at the top, and 'SIGILLUM BOLOGNENSIS' is at the bottom. The seal is centered on the page.

Broadcasting at the Critical Threshold in Peer-to-Peer Networks

Stefano Arteconi

David Hales

Ozalp Babaoglu

Technical Report UBLCS-2006-22

October 2006 - Revised March 2007

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)

The University of Bologna Department of Computer Science Research Technical Reports are available in PDF and gzipped PostScript formats via anonymous FTP from the area `ftp.cs.unibo.it:/pub/TR/UBLCS` or via WWW at URL `http://www.cs.unibo.it/en/research/reports/`. Plain-text abstracts organized by year are available in the directory ABSTRACTS.

Recent Titles from the UBLCS Technical Report Series

- 2005-21 *Greedy Cheating Liars and the Fools Who Believe Them*, Arteconi, S., Hales, D., December 2005.
- 2006-01 *Lambda-Types on the Lambda-Calculus with Abbreviations: a Certified Specification*, Guidi, F., January 2006.
- 2006-02 *On the Quality-Based Evaluation and Selection of Grid Services (PhD Thesis)*, Andreozzi, S., April 2006.
- 2006-03 *Transactional Aspects in Coordination and Composition of Web Services (PhD Thesis)*, Bocchi, L., April 2006.
- 2006-04 *Semantic Frameworks for Implicit Computational Complexity (PhD Thesis)*, dal Lago, U., April 2006.
- 2006-05 *Fault Tolerant Knowledge Level Inter-Agent Communication in Open Multi-Agent Systems (PhD Thesis)*, Dragoni, N., April 2006.
- 2006-06 *Middleware Services for Dynamic Clustering of Application Servers (PhD Thesis)*, Lodi, G., April 2006.
- 2006-07 *Meta Model Management for (Semi) Structured and Uncertain Models (PhD Thesis)*, Magnani, M., April 2006.
- 2006-08 *Towards Abstractions for Web Services Composition (PhD Thesis)*, Mazzara, M., April 2006.
- 2006-09 *Global Computing: an Analysis of Trust and Wireless Communications (PhD Thesis)*, Mezzetti, N., April 2006.
- 2006-10 *Fast and Fair Event Delivery in Large Scale Online Games over Heterogeneous Networks (PhD Thesis)*, Palazzi, C.E., April 2006.
- 2006-11 *Interoperability of Annotation Languages in Semantic Web Applications Design (PhD Thesis)*, Presutti, V., April 2006.
- 2006-12 *Advanced Machine Learning Techniques for Digital Mammography (PhD Thesis)*, Roffilli, M., April 2006.
- 2006-13 *Modular Algorithms for Component Replication (PhD Thesis)*, Vuckovic, J., April 2006.
- 2006-14 *Performative Patterns for Designing Verifiable ACLs*, Dragoni, N., Gaspari, M., April 2006.
- 2006-15 *Towards Flexible Information Systems: Bridging the Gap Between Theory and Practice*, Magnani, M., Montesi, D., May 2006.
- 2006-16 *Improving the Selection of Close-Native Protein Structures in Decoy Sets Using a Graph Theory-Based Approach*, Casadio, R., Fariselli, P., Margara, L., Medri, F., Vassura, M., May 2006.
- 2006-17 *A Secure Peer Sampling Service as a Hub attack countermeasure*, Jesi, G. P., Gavidia, D., Gamage, C., van Steen, M.
- 2006-18 *Modified Realizability and Inductive Types*, Asperti, A.; Tassi, E., June 2006
- 2006-19 *Towards Automatic Social Bootstrapping of Peer-to-Peer Protocols*, Hales, D.; Babaoglu, O., June 2006
- 2006-20 *SOCK: a calculus for service oriented computing*. Guidi, C.; Lucchi, R.; Zavattaro, G.; Busi, N.; Gorrieri, R., July 2006
- 2006-21 *Evolving Networks for Social Optima in the "Weakest Link Game"*, Rossi, G.; Arteconi, S.; Hales, D., July 2006

Broadcasting at the Critical Threshold in Peer-to-Peer Networks¹

Stefano Arteconi²

David Hales²

Ozalp Babaoglu²

Technical Report UBLCS-2006-22

October 2006 - Revised March 2007

Abstract

Peer-to-peer (P2P) applications often require periodic broadcasting of messages from a single node to the entire network. For example, a node may wish to inform the network the availability of a new service or resource, or a node may want to query the network for a particular file or service. In unstructured P2P networks, a simple solution to this problem is to use a “flood-fill” algorithm. In flood-filling, each node passes the message to all of its neighbors in the network when it first receives it. Although this approach is robust to high node turnover and failure rates, it is highly inefficient since many more messages than necessary are sent. Furthermore, the approach is vulnerable to “free-riding” nodes that may receive messages but do not want to invest their resources (bandwidth) for forwarding them to neighbors. In this paper we present a modified flood-fill algorithm that increases efficiency and reduces the incentive for free-riding while retaining the robustness and simplicity of the original scheme. Our algorithm makes use of an evolutionary approach in which nodes copy strategies of others. Interestingly, the technique self-organizes the network towards a critical threshold of the network. The approach is novel and may have applications beyond broadcasting.

1. Partially supported by the EU within the 6th Framework Programme under contract 001907 Dynamically Evolving, Large Scale Information Systems (DELIS).

2. Dept. of Computer Science, The University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy.

1 Introduction

Many P2P applications require a general broadcasting service that will propagate a message from any node to all other nodes in the network. For example, file-sharing systems such as gnutella [9] require queries from nodes to be disseminated to the entire network. P2P networks vary in their topologies: some maintain structured topologies while others choose unstructured and highly dynamic schemes.

In this paper we examine broadcasting in unstructured and possibly dynamic P2P overlay networks. Overlay networks are logical topologies constructed over existing infrastructures such as the Internet. Each node maintains some small number of logical links to a set of neighbor nodes. An Internet-based P2P overlay, for example, may store a set of IP addresses at each node indicating those nodes that are known, and thus, can be communicated with.

A simple idea for implementing a broadcast function would be for each node to pass all new broadcast messages it receives to all other nodes it knows about (it's neighbors). In this manner, a message initiated from any node will eventually reach all other connected nodes in the network. This technique is termed flood-fill (FF) and despite being simple and robust, it can be very costly in terms of messages: FF will send a number of messages equal to the number of links in the network (assuming the network is connected).

The optimal broadcasting strategy in terms of communication cost is for the message to follow a spanning tree such that each node passes the message only to those neighbors that have not already received the message. This approach requires a number of messages equal to the total number of nodes in the network.

Given a static network, a spanning tree can be constructed for a given source node through an initial FF broadcast where node information concerning message spread is retained. Hence, for certain environments where the network is relatively static and some small number of source nodes initiate broadcasts, this approach may be practical.

In many P2P applications, however, the network is typically very dynamic and the number of nodes that need to initiate a broadcast may be very large. A further issue in many P2P systems is that they are open in nature, which means that there is no central administrative control to ensure that all nodes follow a given protocol. Anyone can modify and release peer clients that may exhibit selfish or malicious behavior. In the context of broadcasting, a node may gladly receive messages but may not pass them on to its neighbors since this might incur costs such as bandwidth and power consumption. Given these factors, many P2P designers opt for a pure FF approach.

The contribution of this paper is twofold. First, we present a new broadcast algorithm where nodes dynamically decide whether to pass on messages based on a simple adaptive evolutionary protocol. Through simulation studies, we observe that despite selfish behavior of nodes, the approach produces reasonable broadcast performance with a total message cost that is significantly less than the FF approach. As a second contribution, our studies reveal that the evolutionary broadcast algorithm evolves the network towards a critical threshold that could be related to site-percolation theory.

In related work on this topic, broadcasting applications have been optimized through *a priori* analysis of the site-percolation critical threshold [13]. In these studies, the threshold value, which is pre-computed, is later used as a parameter of the broadcast protocol. Our approach differs from these in that there is no pre-computation step and the network itself self-organizes through simple, local interactions between nodes, which behave myopically and selfishly, copying the behavior of other nodes that are achieving higher utilities.

The rest of this paper is organized as follows: In section 2 a broadcasting scenario is defined as a game and a simple evolutionary rule by which nodes update their behavior is described. Simulation results are then presented and discussed in section 3 and a possible interpretation is given in section 4. Finally, related work is presented in section 5 and we draw our conclusion in section 6.

```

j ← randomNode ()
if ( Utilityj > Utilityi ) then
  Si ← Sj
  with probability  $\mu$  change Si
  Utilityi ← 0

```

Figure 1. The evolutionary state update algorithm.

2 The Broadcast Scenario

For simulation purposes, we assume that the overlay network is a random graph. We study such networks under both static and dynamic conditions. Each node in the network adopts one of two states or strategies: PASS or DROP. A PASS node always passes a newly received broadcast message to all of its neighbors (excluding the one from which it received the message). A DROP node is a “sink” and does not pass any of the messages it receives.

2.1 A Broadcasting Application

We cast the problem in the form of a “broadcast game” where nodes are awarded a payoff based on the outcome of each broadcast event. A broadcast is initiated by randomly selecting a node as the source of the broadcast which passes the new message to each of its neighbors. In the next time-step, each neighbor in turn decides whether to further pass the message or not based on its current strategy. This process continues until no more messages can be sent.

Intuitively, nodes desire to receive broadcast messages but they have an incentive not to pass messages to others. Hence, we assume there is some benefit B to be gained by receiving a message and some cost C to be paid for sending a message. We assume that $B > C > 0$. For each broadcast event, nodes in the network receive a payoff equal to B if they received but did not pass the message, $B - C$ if they received and passed on the message, or 0 if it did not receive the message.

2.2 The Evolutionary Algorithm

A cycle denotes the period of time during which all nodes are activated in some random order. When a node is activated, it checks its input buffer for a new message. If a message is found, the node will forward it to all its neighbors if it is operating under a PASS strategy, or does nothing if it is operating under a DROP strategy. Then, the node receives a payoff as described above. A running average of the payoffs that each node receives is computed and stored locally as the node’s current *utility*.

After message handling, each node executes the *strategy update* algorithm depicted in Figure 1 with some probability ρ . The pseudocode shown is for node i where $Utility_i$ and $S_i \in \{\text{PASS}, \text{DROP}\}$ denote its current utility and strategy, respectively. In this algorithm, the executing node i selects some other peer j randomly from the entire population³ and compares its utility with that of j . If peer j has a higher utility, then node i copies its strategy, in which case node i ’s strategy undergoes *mutation* with some small probability μ . Mutation results in the strategy flipping to the other state (since there are only two strategies). In this manner, nodes receiving higher utility will tend spread their state to nodes with lower utility through what can be seen as “replication”. Mutation, on the other hand, allows spontaneous switching of strategies.

3 Simulation Results

To analyze the performance of the system, three measures were calculated:

3. A peer sampling service is required to implement `randomNode()`. In our simulations we use the NEWSCAST [10] protocol for implementing a fully distributed random peer sampling service.

- *PASS*: Proportion of nodes adopting the *PASS* strategy.
- *MC* (Message Count): Number of copies of any single message that are sent.
- *NR* (Number of Receptions): Proportion of nodes receiving a message.

Obviously *PASS* and *MC* are related because the more nodes that forward received messages, the more message copies are sent. On the other hand to obtain a high value of *NR* it is not necessarily required to have a high *PASS* value. A high *PASS* value translates into high redundancy in terms of many messages being sent to nodes that have already received them.

Our objective is to have the highest possible *NR* ($NR = 1$, meaning a message is seen by all nodes) while keeping *PASS* and *MC* values as low as possible. In other words we want as many nodes as possible to be able to receive the broadcasted messages while minimizing the number of message copies sent.

In the rest of this section simulation results concerning both fixed and dynamic networks of different sizes are shown and discussed. In the following section we propose an interpretation consistent with the results based on a threshold for the number of nodes forwarding messages.

3.1 Parameter Values

Our simulations were carried out in a network of size $N = 4000$, with each node having at most 20 neighbor links. The payoffs values were $B = 1$ and $C = 0.25$ resulting in the payoff matrix illustrated in Table 1⁴. The replication probability was set to $\rho = 0.2$ and mutation rate $\mu = 0.005$. A new broadcast event was initiated every 20 cycles. Note that this value is sufficiently large such that a message propagates throughout the network before a new broadcast is initiated.

	PASS	DROP
Received	0.75	1
Not Received	0	0

Table 1. Payoff matrix. Nodes always benefit by selecting the DROP strategy since they avoid the message forwarding cost. A node which does not receive a message always gets payoff 0. A large percentage of DROP nodes may prevent the message from spreading through the network.

The broadcast application has been tested both on fixed topology networks and on dynamic networks where nodes are periodically added and removed in order to simulate the *churn* phenomenon typical of P2P networks.

3.2 Fixed Topology

To generate the random topology, the network is initially composed of N nodes without any edges. To define the edges, we pick one node at a time and add links to 20 (recall that this is the viewsize limit) randomly picked destination nodes. If the destination node already has a full view, then the link is not added. This method leads to a random topology where most of the nodes have a full view while some nodes (the last ones picked for adding links) are not able to fill their view due to the difficulty of finding random nodes whose views are not complete yet.

Figure 2 shows the typical trend of a single run of the algorithm on a 4000-node network. All of the node strategies are initialized to *PASS* (hence initially a flood fill is performed) and a new message is broadcast by a randomly picked node every 20 simulation cycles.

At the beginning of the simulation, *PASS*, which is initially 100%, starts to drop while *NR* remains very high. This indicates that not all nodes need to follow the *PASS* strategy for the message to propagate throughout the entire network.

4. Experiments with different C values have been performed leading to results showing negligible difference with the cases illustrated and analyzed here ($C = 0.25$).

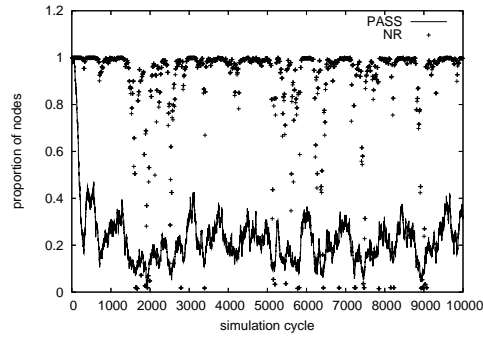


Figure 2. Single run of the broadcast application on a 4000-node static network. The solid line represents proportion of nodes adopting PASS behaviour (*PASS*), while each cross represents the proportion of nodes that receive a given message.

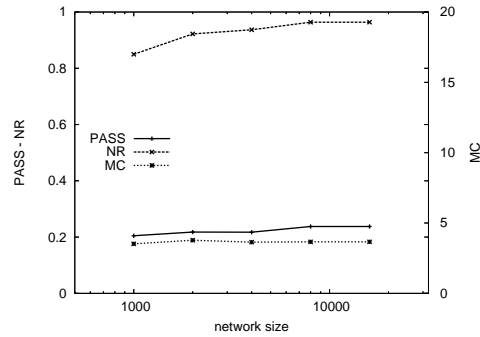


Figure 3. Performance on static networks of different sizes (1000, 2000, 4000, 8000, 16000 nodes).

Eventually, *PASS* drops too low to guarantee the message from reaching all destinations, at which point *NR* starts to drop (around cycle 1500 in the example shown in figure 2). Notice how at this point *PASS* stops dropping and begins to oscillate.

It appears that the system lowers *PASS* as long as there is some level of redundancy. When the number of nodes are operating with the *PASS* strategy falls below a critical threshold necessary for network-wide message propagation, *PASS* oscillates around that value.

Throughout the simulation, there are sudden dips (or dropouts) in *NR* indicating that certain messages reach a very low number of nodes. This could happen by chance if a node surrounded by *DROP* nodes is chosen as a source for a broadcast event, or as a result of a *PASS* value being too low to sustain good message spreading. The system reacts well in both cases, in fact while occasional dropouts due to an unfortunate source node selection do not affect the *PASS* trend, continuous dropouts due to a too low *PASS* value cause it to increase, bringing *NR* back to higher values as a consequence.

In Figure 3 algorithm performance related to different network sizes ranging from $N = 1000$ to $N = 16000$ nodes is shown. Each point is the average value of 5000 distinct broadcasts (10 different runs of 10000 cycles).

Examining *NR*, we note that it remains relatively high (above 0.8) for every network size and that it benefits from increased network size, approaching almost 100% spreading in 16000-

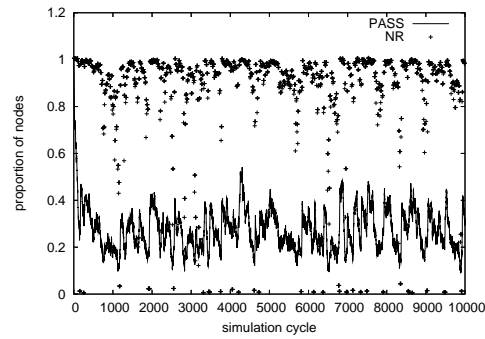


Figure 4. Single run of the broadcast application on a 4000-node dynamic network (1% of the nodes substituted in each cycle). Compared to the static case *PASS* is higher but message spreading drops more often.

node networks. Although some nodes fail to receive messages, it is important to note that since node strategies are continuously changing it is unlikely that the same nodes will repeatedly fail to receive messages. Moreover, notice that *NR* improves in bigger networks. This is a highly desired property in P2P networks which can be extremely large. The system also scales very well for *PASS* and *MC* as their values remain almost constant over the range of network sizes studied.

3.3 Networks with Churn

To evaluate network performance in presence of churn (dynamicity introduced by nodes continuously leaving and entering the system) we ran experiments with the same settings described in the static case but at each cycle we removed 1% of the nodes chosen randomly and reintroduced the same number as new nodes. The new nodes are linked randomly to preexisting nodes and set to a random strategy chosen equally between *PASS* and *DROP*.

As can be seen from Figure 4, the effect of churn is twofold: *PASS* oscillates around higher values than in the static case and *NR* drops more often.

The high *PASS* value is a result of the joining nodes adopting a *PASS* strategy with probability 0.5 (because the system usually stabilizes at a lower value). As for the lower *NR* value, it could be explained as a consequence of nodes leaving the network because when a node is removed its incoming message queue containing messages to be forwarded at the next cycle is lost.

As can be seen in Figure 5, even though *NR* remains high in dynamic networks (unlike the static case), its value decreases when the network grows. This could be an effect of the relative growth of the churn rate with respect to the message spreading speed. While the churn rate is always set to 1% relative to the network size, the message spreading speed is bounded by the network topology, specifically by the maximum viewsize which is fixed independently from network size. We note that the churn rate we adopted to test the network is extremely high: basically 1% of the nodes enter and leave the network in the time needed by a node to send a message to a neighbor. Most likely, the network would scale better with lower churn rates.

4 A Critical Threshold

The results presented in the previous section may initially appear puzzling. Given that nodes always have an incentive to use a *DROP* strategy why would not all nodes end up using a *DROP* strategy? Also, why does *PASS* oscillate around a value that appears to be a critical threshold?

We believe that the process may operate as a sort of self-organizing evolutionary process around the critical threshold for the network. We summarize the process in the following way:

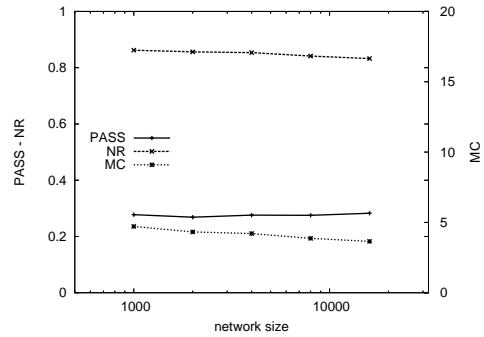


Figure 5. Performance on dynamic networks of 1000, 2000, 4000, 8000 and 16000 nodes subject to 1% churn.

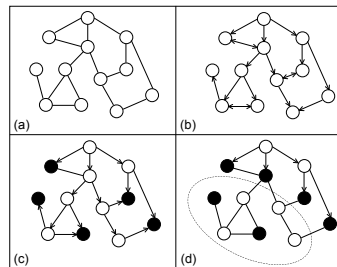


Figure 6. Broadcasting on a simple graph with PASS and DROP nodes. (a) shows a graph representing the network. (b) shows the route messages take through the network assuming that the uppermost node initiates a broadcast. (c) shows the same graph with some DROP nodes in black, notice that in this case all nodes still receive the message. (d) shows a situation in which a portion of the nodes (inside the dotted region) do not receive the message.

- When *PASS* is high there are many redundant messages being sent.
- Nodes can adopt DROP behaviors lowering network traffic without affecting *NR*.
- Eventually the *PASS* value falls below some critical threshold (k) causing the formation of clusters of nodes unable to receive messages.
- Both *PASS* and DROP nodes receiving messages are equally likely to be copied by nodes that don't get any message.
- Hence the *PASS* value found in the cluster of nodes receiving the message would tend to be reproduced by nodes unable to receive messages (having utility 0).
- *PASS* nodes receiving the message will tend to copy the DROP behaviour of nodes receiving the message hence breaking down clusters that receive the message over time.
- Such cyclic behaviour causes the oscillation of *PASS* value around a threshold value.

In summary, when *PASS* falls below a critical threshold, nodes can be divided into a connected cluster around the source broadcast node that receive the message and those in disconnected clusters that don't receive the message. Given the payoffs, the disconnected nodes will converge to zero utility over time and will tend to reproduce the proportion of *PASS* strategies from the connected cluster where all nodes have non-zero utility. This proportion should be around the critical threshold at the point the network disconnects. Figure 6 illustrates this notion with a small graph (a) and the route messages would take if sourced from the uppermost node if all nodes used a *PASS* strategy (b). Note that in (c) a number of nodes use a DROP strategy but all nodes still get the message. However, in (d) a region of the network does not receive the message.

4.1 Simulation Experiment

To check if the value of $PASS = 0.2$ corresponds to a critical threshold under below the network becomes disconnected, we tried a very simple simulation experiment. Many fixed random networks were generated with the algorithm described in section 3.2, then nodes were removed progressively (a removed node corresponds to a node with DROP behaviour) and the size of the biggest component, analogous to the portion of the network that is able to receive messages was calculated.

As shown in Figure 7, these experimental results suggest that the 4000-node network begins to partition when about 3200 nodes have been removed. The remaining 800 nodes correspond exactly to 20% of the initial size, confirming our hypothesis for explaining the critical threshold value in the broadcast application.

4.2 Possible Link to Percolation Theory

The value reached by *PASS* oscillates around the lowest value in which the network is still able to send messages. To put it another way, it is the minimum number of *PASS* nodes needed for the network not to be partitioned. This description is very similar to *percolation threshold* [3, 12].

In percolation theory, each site in a graph (node in a network) can be either active or inactive. The percolation threshold of a graph is a phase transition point indicating the minimum proportion of nodes that should be active to ensure that they form a unique connected cluster. While for simple topologies, such as square lattices, exact threshold values are known, for more complex topologies only analytical models exist (see for example [3, 12]).

Unfortunately, in our system we are not only interested in *PASS* nodes being part of a single cluster, but we want messages to be spread throughout the network. In order to obtain this, we also need each DROP node to be connected to at least one *PASS* node. This additional constraint makes the problem we are interested in similar to the *vertex cover* problem [8]. A vertex covering for a given graph G is defined as the set of vertices V such that every edge of G is incident to at least one vertex in V . Finding the minimum vertex cover in a graph is an NP-complete problem. The threshold found through simulations in our system seems to share properties both

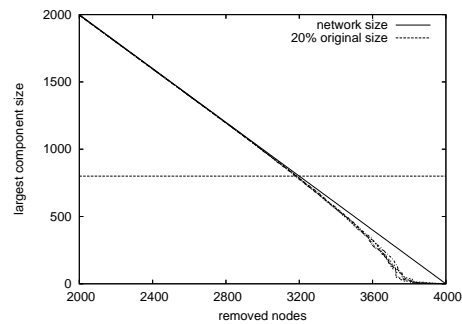


Figure 7. Determining when networks become disconnected. At each cycle a random node is removed. The black line represent the total network size the dotted lines represents the size of the largest components in different runs. The horizontal dotted line indicates 20% of the initial size and corresponds to a point where the largest component is lower than the network size, indicating network partitioning.

with vertex cover in the sense that we want each node (even DROP ones) to receive messages, hence to be reached by at least one active link, and with percolation threshold for the reason that PASS nodes should be connected so to play the role of a *backbone* through which messages are spread throughout the network.

Percolation threshold and the vertex cover in random networks are hard problems, but the possible relations between these two well-known and studied problems and our simulated system increases our confidence for the possibility to give an analytical solution of our broadcast model and on the goodness of the threshold value adaptively reached through a simple evolutionary scheme.

5 Related Work

The evolutionary scheme based on strategy copying of better performing nodes is drawn directly from the SLACER algorithm described in [6], however the original algorithm involves link rewiring in addition to strategy copying leading to a highly dynamic small-world like networks. Being interested in information dissemination through broadcasting we decided to eliminate rewiring to avoid a highly clustered topology that would influence negatively broadcast performance.

The SLACER algorithm has been shown to possess good adaptivity properties when used in different environments such as producing cooperation in Prisoners' Dilemma game [6] and in a simple file-sharing model [7], and to be resilient to the presence of cheating nodes subverting the assumption that nodes honestly report their strategy and utility when comparing to other nodes [2].

The versatility and robustness shown so far by the SLACER algorithm pushed us to use a modified version (with no rewiring) to tackle the problem of optimized broadcasting.

Broadcasting optimization is a central problem in distributed systems and many solutions have been proposed. In P2P networks several approaches based on gossip protocols exist to perform probabilistic broadcast on a network [4] or to diffuse and aggregate information [11].

Finding optimal ways to diffuse information is highly relevant in ad hoc networks, particularly in the study of vehicular networks where information about traffic are to be exchanged by traveling vehicles and many restrictions and problems related to the radio medium (limited transmission range, collision and interference) need to be faced. Among the proposed techniques the adaptation of transmission rate [14] or retransmission probability and delay [1] bare some similarity to our approach.

The rapidly growing knowledge of complex network structures and properties led in relatively recent times to the use of techniques from many disciplines to the study of network properties and algorithms. Percolation theory is widely used to design or analyze network algorithms, especially when facing problems of information dissemination or network connectivity. A direct application of percolation theory to the design of an optimized broadcast for ad hoc networks is proposed in [13], while an analysis of wireless networks properties based on percolation theory is given in [5].

Percolation theory and network structure has been combined to study important properties such as cluster size and distribution, and connectivity probability in random [3] and small-world [12] networks.

We do not know of any previous work that self-organises nodes towards a critical threshold using a population level evolutionary algorithm utilizing the mechanism we identify in this paper.

6 Conclusions

In this paper we have shown how a simple algorithm running in the nodes (a P2P protocol) can self-organize a network around a critical threshold using an evolutionary approach within a broadcasting scenario. Even though nodes have an incentive in the scenario to drop messages the systems keeps the number of nodes which are forwarding messages high enough to allow most of the network to receive messages. This approach is scalable and robust to churn (highly dynamic networks). This is surprising since nodes act selfishly in a bounded way — they copy other nodes with higher utility and each node has an incentive not to pass on messages.

The process which keeps the number of passing nodes from dropping permanently below the critical threshold appears novel and interesting since it to exploits a global property (the network becoming disconnected) which feeds-back to the local behavior (evolution based on individual utility) to give a desirable self-tuning result.

There are still open issues which need to be addressed before using our approach in a P2P protocol. The network does not settle and stabilize at the critical threshold but oscillates widely around it. Also we assume honest reporting of strategies and utilities and this may not be the case with malicious or incorrectly functioning nodes. Although the number of messages sent for broadcasting appears to be lower than a flood-fill approach, we have not factored in the message overheads required for communicating strategies and utilities. This could undermine the advantages.

The technique we propose exhibits many interesting properties, such as self-organized criticality in adapting the number of passing nodes on a limit value needed to spread messages throughout the whole network, good scalability and resilience to churn. The possibility to relate our approach to well-known and established problems in percolation theory or connected cover sets could make it possible to perform further analytical investigation for the the design of additional algorithm features and optimization. Even though an analytical approach seems to be feasible, much more experimental study should be performed. In particular, the system's sensitivity to network topology and to evolutionary parameters, such as mutation rate and replication probability have to be further investigated⁵.

Although our interest is derived from a P2P engineering perspective, perhaps the evolutionary mechanism demonstrated could be applicable to other physical phenomena observable in the world.

7 Acknowledgements

We would like to thank Mark Jelasyty for discussions that have influenced the paper.

5. Indeed, recent experiments show that the results are sensitive to the replication rate used by nodes. This is the subject of on-going work.

References

- [1] H. Alshair and E. Horlait. An optimized adaptive broadcast scheme for inter-vehicle communication. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 5, pages 2840–2844, 2005.
- [2] S. Arteconi, D. Hales, and O. Babaoglu. Greedy cheating liars and the fools who believe them. In *Engineering Self-Organising Systems*, volume 4335 of *Lecture Notes in Computer Science*. Springer, 2007.
- [3] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85:5468, 2000.
- [4] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. In *DSN '01: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS)*, pages 443–452, Washington, DC, USA, 2001. IEEE Computer Society.
- [5] M. Franceschetti, L. Booth, M. Cook, R. Meester, and J. Bruck. Percolation of multi-hop wireless networks. Technical Report UCB/ERL M03/18, EECS Department, University of California, Berkeley, 2003.
- [6] D. Hales and S. Arteconi. Slacer: A self-organizing protocol for coordination in peer-to-peer networks. *IEEE Intelligent Systems*, 21(2):29–35, Mar/Apr 2006.
- [7] D. Hales and B. Edmonds. Applying a socially-inspired technique (tags) to improve cooperation in p2p networks. *IEEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans*, 35(3):385–395, 2005.
- [8] http://en.wikipedia.org/wiki/Vertex_cover_problem. Date examined: 14/02/2007.
- [9] <http://www.gnutella.com>. Date examined: 14/02/2007.
- [10] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In H.-A. Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 79–98. Springer-Verlag, 2004.
- [11] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.
- [12] C. Moore and M. Newman. Exact solution of site and bond percolation on small-world networks. *Physical Review*, 62:7059–7065, 2000.
- [13] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2003)*, Mar. 2003.
- [14] L. Wischhof, A. Ebner, and H. Rohling. Information dissemination in self-organizing inter-vehicle networks. In *IEEE Transactions on intelligent transportation systems*, volume 1, pages 90–101, 2005.