The role of complex systems in the management of pervasive computing

Ian W. Marshall*, Chris Roadknight* and Lionel Sacks^
*BTexact Technologies, B54/124, Adastral Park, Martlesham Heath, Ipswich, Suffolk, UK. IP5 7RE
^University College London

Abstract.
Network complexity will increase dramatically over the next 5 years as will the amount of devices inhabiting these networks. Ad-hoc and active paradigms will make the already onerous task of network management increasingly problematic. An approach to managing such networks based on bacterial colony behaviour is discussed, offering innate abilities for essential tasks such as software proliferation, load balancing and differing but distinct qualities of service. Robustness to fractal request streams is also demonstrated using real world requests as a source of simulated network load. The 'hands off' element of the adaptive algorithm is a major asset for any configuration and optimisation task. This biologically inspired adaptive management solution could be the ideal approach to managing the behaviour of complex data networks of the future.

I. Introduction

Extrapolation of current trends for ownership of microprocessors [1] suggests that within 10 years it is possible that many individuals will own in excess of a thousand micro-controllers. If, as seems increasingly likely, pervasive computing on this scale is realized, users will be faced with a major investment of time and money in configuration and maintenance activities. To minimize this impact it is important to investigate ways of automating low level management processes and enabling many pervasive computing devices to self-configure and operate almost autonomously [2]. At the same time it is vital to ensure that the management processes are able to adapt to new requirements and applications, since it is likely that developments will be extremely rapid and unpredictable [3].

Clearly any management system that can meet these requirements is going to exhibit dissipative structures, and long-range dependency (leading to scale free properties and fractal dimension), much like many natural systems. We have simulated the operation of a management system (inspired by a simple model of biofilm colonies) that autonomously configures and maintains a network of up to 5000 heterogeneous devices. A key feature is the ability of nodes in the system to perform unsupervised in-situ learning by exchanging policies with one another. This was achieved by combining policy-based management [4] with an evolutionary algorithm derived from prokaryote biology [5]. The algorithm combines rapid learning (plasmid exchange), automated configuration (limited motility of individuals) and evolutionary learning via a conventional mutation based GA. New policies can be evolved internally. In addition users are able to distribute new policies to subsets of the devices that they control, using a weakly consistent gossip protocol derived from fireflies [6].   Simulations have shown the system is able to create and maintain an organization that performs at least as well as conventional designs such as caching systems, whilst requiring no human input or intervention. In particular the system copes extremely well with multi-fractal load derived from real Internet traffic logs, once it has self-organized into a metastable critical state.

In this paper we motivate and describe a novel proposal for automating the management of future networks, the devices that inhabit them` and the software that will run on them. This adaptive approach is inspired by observations of bacterial communities. We also present results that clearly demonstrate that our proposal can offer 'hands off' management for networks of hardware devices using load scenarios based on mathematical models and real world inspired request streams.

II. Related Approaches

Adaptive control is based on learning and adaptation [7]. The idea is to compensate for lack of detailed knowledge by performing experiments on the system and storing the results. This means monitoring resource demands, selecting policies based on behaviour and then implementing policy changes on a local or global level. This style of control has been proposed for a range of Internet applications including routing [8], security [9, 10]. At present there seems to be no application of adaptive control to network service configuration and management.

At first sight adaptation can seem an unsophisticated option in comparison to classical control methods, which are based on monitoring state, deciding on the management actions required to optimize future state, and enforcing the management actions. Adaptive control is much more human inspired, as it attempt to automate the behaviour of an expert, monitoring the network and making changes based on their interpretation of this monitoring. In classical control the decision is based on a detailed knowledge of how the current state will evolve, and a detailed knowledge of what actions need to be applied to move between any pair of states (the equations of motion for the state space). It is this need for complete and detailed knowledge of all cause/effect relationships that make such method untenable for many real-world, dynamic, non-linear problems. Many control schemes in the current Internet (SNMP, OSPF) are based on this form of control. There is also a less precise version known as stochastic control, where the knowledge takes the form of probability density functions, and statistical predictions. All existing forms of traffic management are based on stochastic control, typically assuming Poisson statistics.

Genetic Algorithms (GAs) have been shown to offer a robust approach to evolving effective adaptive control solutions [11]. Recently many authors [12, 13,14] have demonstrated the effectiveness of distributed GAs using an unbounded gene pool and based on local action (as would be required in a multi-owner internetwork). However, many authors, starting with Ackley and Littman [15], have demonstrated that to obtain optimal solutions in an environment where significant changes are likely within a generation or two, the slow learning in GAs based on mutation and inheritance needs to be supplemented by an additional rapid learning mechanism. Harvey [16] pointed out that gene interchange (as observed in bacteria [17, 18]) could provide the rapid learning required. This was also demonstrated for a bounded, globally optimised GA [19]. In this paper we demonstrate that an unbounded, distributed GA with "bacterial learning" is an effective adaptive control algorithm for many aspects of an active service provision system derived from the application layer active network (ALAN) [20, 21, 22].

There is research on schemes for load balancing and resource management on servers that offers some comparisons. One such work looks at adaptive algorithms for load balancing over a cluster of web servers [23]. Here it is proposed that the TTL's of DNS address mappings are calculated adaptively, taking in to account the distribution of client requests and the heterogeneity of the web servers.

Another method of service allocation control involves setting thresholds for acceptance/rejection based on hardware requirements and a priority class reward system [24]. Server capacity is divided up virtually with priority threshold values for each virtual partition. While the setting of dynamic thresholds is reward based and hence has some adaptive features, it does not deal with mobile software and the policies acting upon this software.

III. Simulation Design

The simulation takes the form of a cellular automata [25] where each cell behaves as a simple organism fighting for survival against other similar organisms in an environment of changing 'resources'. All interactions take place on a local basis, with each cell only communicating with a few other 'nearby' cells and there being a large number of cells. There is an important difference in comparison to standard genetic algorithms in that in this case a fit colony is the ultimate goal, not one superfit individual. The organisms that we chose to base our automata on were bacteria, there are several reasons for this choice, bacteria have novel reproduction methods, including plasmid migration, that mean they evolve very quickly but with a robustness that has kept many varieties present for several billions of years [26, 18]. Plasmid migration means sections of genome can be transferred between living organisms effecting their phenotype in a much more direct way than Darwinian methods. They are also very simple organisms whose genetic structure is completely known.

The model presented here randomly places automata, with random initial genomes, at some of the vertices of a 2-dimensional grid that forms their environment. The grid structure aids the understanding of the model, but a simple grid like structure is not a pre-requisite. The genome of each bacteria-like automata contain plasmids that enable the host to metabolise different kinds of available nutrients, and earn a reward (energy) for each molecule metabolised. For the purposes of this research a plasmid is the specific software that requires a certain nutrient plus a set of subjective rules as to whether the nutrient should be accepted (eg. Software A will be run if the unit is less than X% busy and has a queue smaller than Y. X and Y being initially randomly generated but are subject to subsequent evolutionary selective pressure). There are up to 100 different nutrient molecules. A random subset of these molecules is injected at each vertex along one side of the grid, at the beginning of every epoch (an epoch is 100 timesteps or 1 hop). The size of the injected subset is a variable of the model, and can frequently be close to zero. Once injected, nutrient molecules are forwarded (away from or parallel to the side they entered) automatically from vertex to vertex until a vertex is reached with a resident ‘bacterium’. If a suitable gene is present, and the bacterium has not already got enough nutrients, the molecule is metabolized. Otherwise it is forwarded after a small, specified delay. Each movement from one vertex to another is referred to as a hop. The bacteria can maintain a ’queue’ of up to 100 nutrient molecules before they are forced to forward nutrients. If a lot of molecules are decaying in the queue the bacterium can move one place up the grid if there is a vacant vertex. If no molecules are decaying the bacterium can move one place down the grid if there is a vacant vertex. They communicate with neighbours immediately above and below, so swaps are also possible. The time it takes to process 4 molecules is referred to as an epoch. The colony evolves through the exchange and mutation of plasmids. Bacteria that accumulate sufficient energy replicate. Bacteria that do not accumulate energy die. It is possible for some individuals to survive without ever replicating, but they must do some metabolic processing, since energy is deducted each cycle (to simulate respiration).

Nutrient molecules are forwarded with some randomised lateral movement, but when forwarding would take the nutrient beyond the bottom of a sub-colony of microbes the nutrient is forwarded to the top of another sub-colony. Molecules are forwarded to the next OR the next but one layer. These changes enable us to spread the model across several physical computers and increase the size of colony that can be modeled real time, since links between the concentrations are less rich and slightly slower than within concentrations. Our test topology had 6 semi-independent sub-colonies. The 6 sub-colonies are connected in a star formation (1 forwards to 2 and 4, 2 forwards to 1 and 5, 3 forwards to 4 and 6, 4 forwards to 3 and 1, 5 forwards to 6 and 2 and 6 forwards to 5 and 3). Each sub-colony runs on identical hardware and has the same available resources at it's disposal.

To allow the automata to be truly autonomous, as they would be in a real colony, randomly chosen plasmids attach to molecules that are being forwarded, in line with the following rules.

1. If an individual FAILS to process a molecule (and so must forward it) AND a random number between 0-100 is less than the no. of nutrient molecules it has already stored for later processing (max 100) THEN a 'plasmid' from it's genome is attached to the request. In simpler terms; if do not process this item but I AM busy then attach a plasmid from my genome to the item.

2. If an individual succeeds OR fails to process a food item AND its busyness (plus a constant, currently 5%) is less than a random number between 0-100 THEN take a 'plasmid' from the 'plasmid attachment' space, if there is one. In simpler terms; regardless of if I process the item or not, if I am NOT BUSY then take a 'plasmid' from the request if there is one.

This explains the fundamentals of this biologically inspired algorithm but the analogy to networks needs more explanation. Each organism representing a node (or virtual node) on a future multi-service communication network. Each node having the capability (software) to process one or more types of service request in accordance to policies attached to the node or software component. The activity and performance of each node is monitored and used both for transfer of software and as a threshold for differential execution of services. The processing priority of a service request can be determined by a service request function, either 'time to live' or the value derived for carrying out the service request. Each nodal policy comprises a service request identifier and a service request criteria that decides if a service is processed.

If the activity indicator at a node reaches a probabilistic threshold it may add software (or a pointer to software) and the set of subjective policies to a request that it is forwarding. If it exceeds a second threshold for a sufficient period of time the whole nodal components can be replicated to a nearby available node. If a node's activity indicator fails to reach a probabilistic threshold then it can import software (and it's subjective policies) from a request, should these be available. If it's activity indicator stays below a certain threshold for an extended period of time all the active software can be deleted making the node available for new software and policies to be installed.

IV. Results

A. Reaction to increase in total network load.

The cluster was loaded with a poisson distributed traffic stream at all 6 nodes. Initially the software distribution and the associated subjective variables are randomly distributed, so there will be a period of software re-distribution and variable optimisation. Then the load was varied over all nodes to the same extent. After the load is increased there is a period of adaptation as new organisms are generated and plasmids are exchanged. As the load gets excessive at greater than 750 requests per that the system begins to break down and drop rates and latency go up significantly. This is equivalent to a load of about 50% of saturation.

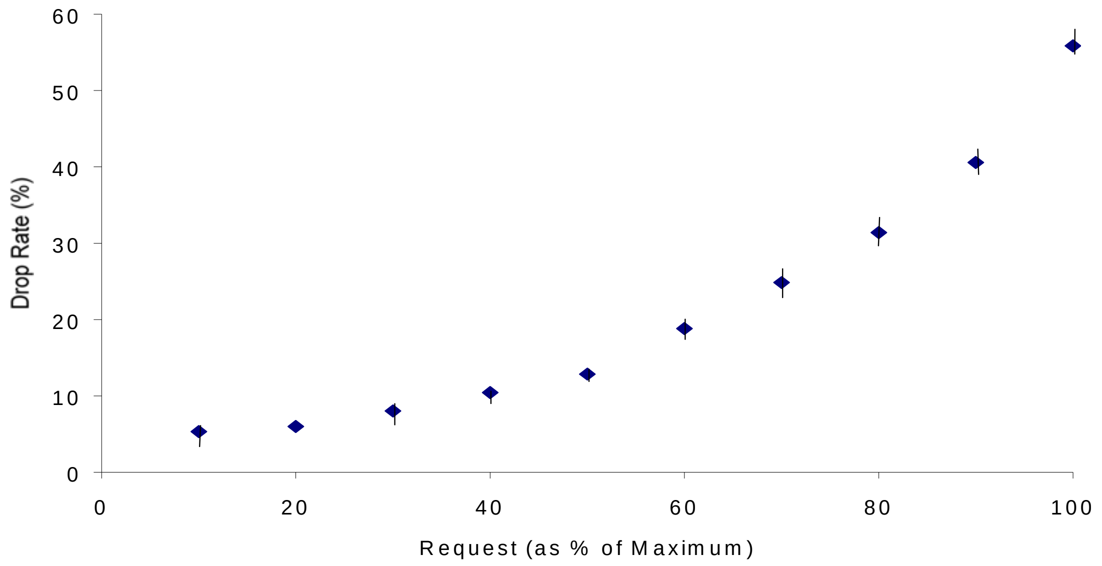

Fig. 1. Drop rate increasing exponentially as load is increased in steps.

The response in the latency of the system is somewhat different (Figure 2.), it increases proportionately at all increases in load. This again is due to the increased queue lengths and time taken to find a suitable node. Lines indicating standard deviations are given, values are generally less than 0.5.

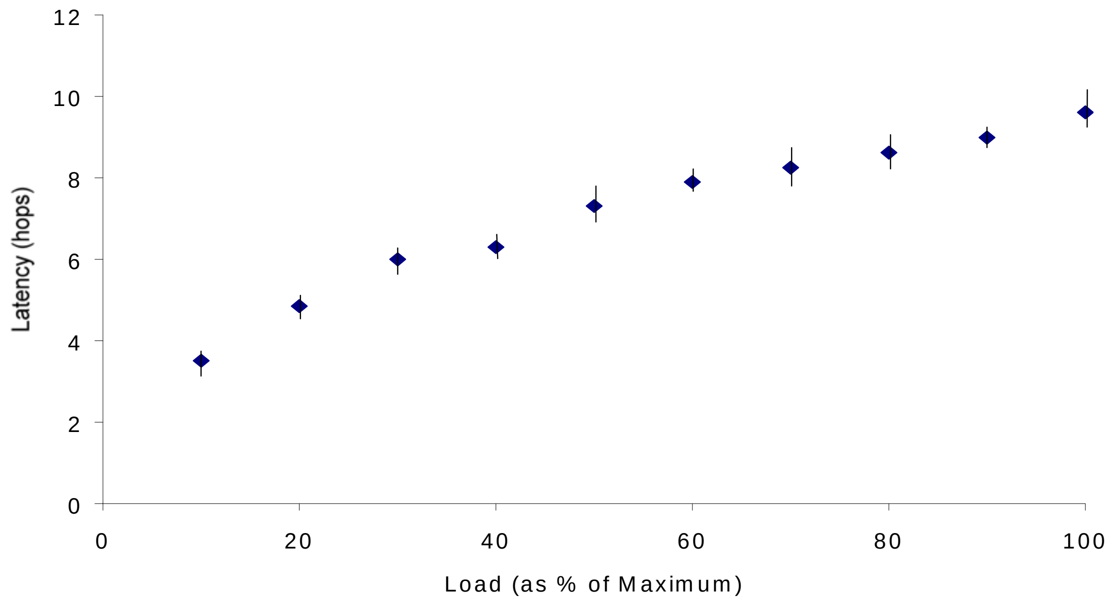

Fig 2. Change in service request latency as load is increased in steps

B. Comparison with other algorithms.

While earlier experiments have compared the bacterial algorithm with itself at different conditions it is also important to compare it to other possible information dissemination algorithms. Therefore the same problem was tackled by an algorithm that randomly places software and subjective rules around the network (random algorithm), and a algorithm that simulates caching by replicating a rule, in the direction of it's request, each time it is used (caching algorithm).

Internet traffic has very irregular characteristics, showing determinism and long-range dependence at both the packet [27] and application request [28] level. This puts different kinds of stresses on any data network. To do this log files from 6 high level web caches were taken for 7 days. The Hurst parameter for occurrences of requests was 0.911, suggesting a significant amount of self-similarity is present in the request stream. Figure 15 shows the results of running the simulation using this log as the traffic generator.

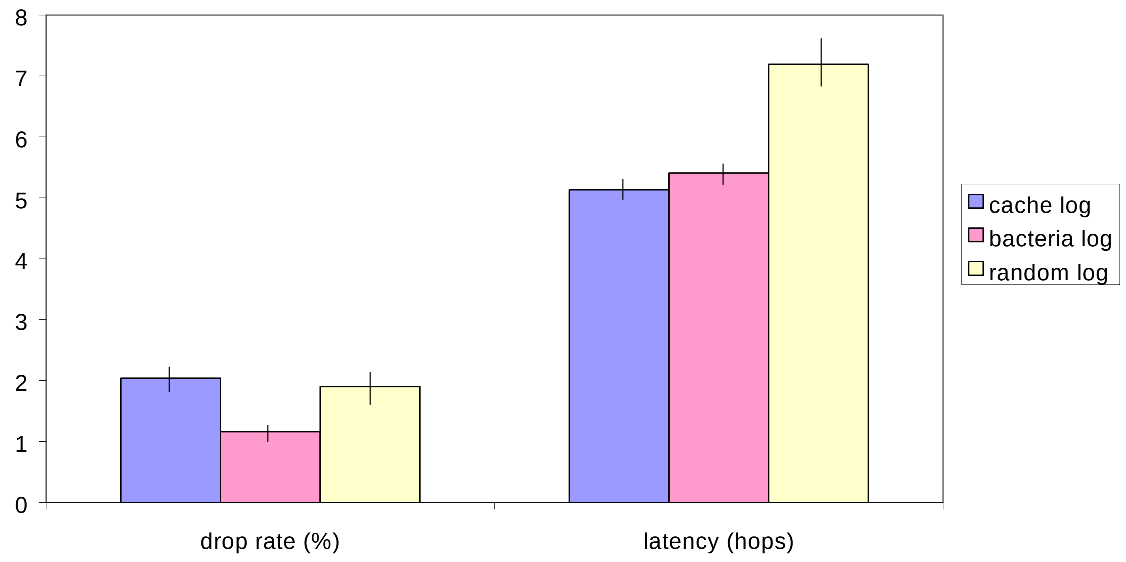

Figure 3. Differences in performance under web request derived load.

It is clear that the bacteria based algorithm outperforms the other 2 algorithms in terms of minimizing the number of requests dropped, though the latency is higher for the bacterial algorithm when compared to the cache based approach. There are 2 reasons for the difference in latency. Firstly the bacterial approach processes more requests per unit time, it is fair to accept that to do this it might take longer to process these requests. More significantly, the adaptive element in this approach means that some hardware nodes in the cluster have many unpopulated vertices, the cache approach uses all vertices. So the bacterial approach is dropping fewer requests while also using less of the hardware resources. It should also be remembered that the strong performance of a bacteria analogy is achieved with the added benefits of increased flexibility and a completely 'hands off' nature.

V. Conclusions

It is apparent that the bacteria inspired algorithm is a suitable choice for providing service completion in a simulated, mobile software enabled network. It has also been shown, in terms of latency, that various discrete qualities of service can be achieved. When compared to conventional resource allocation algorithms (i.e. Caching) it is apparent that a bacterially inspired approach offers many qualitative benefits when a network is loaded in a realistic way. It can be argued that the management algorithm is not designed to optimally handle small numbers of requests for one specific service. Though if a service is unpopular it can be assumed that the user will expect to have to supply code for this service with his requests. The fact that the system offers some facility for these unpopular requests, albeit at a lower standard, is encouraging. Improved performance for the less requested services or at the less loaded nodes could be achieved by using various flavours of quorum sensing [29], for example, an organism could sense the number of organisms with similar phenotypes and be less procreative if there are high numbers of such organisms nearby. A simpler approach could just detect how many neighbours of any kind were present and modifying reproductive thresholds accordingly. But any improvement in the performance of least requested services might be accompanied by a decrease in performance for the popular services (on the 'no free lunch' principle), so the impact of this must be studied.

To provide a practical verification of the simulation results we have embodied the algorithm in an ad-hoc network consisting of 20 nodes, where the nodes are intended to emulate distributed sensor controllers. Each node consists of 3 sensors, 3 actuators, a 16 bit microprocessor and an infra-red transceiver. Initial results are encouraging.

Generalising and proving the capabilities of this system will require complex system models. We are attempting to extend the reaction-transport models of Ortoleva et al [30], since they are a good fit with the capabilities of our current system. We also hope that improved understanding of biogeochemical processes will provide inspiration for extending the functionality of our system.

This work was partly funded by the Royal Society, London

VI. References

[1] L Gerstner, keynote speech at telecom 99, http://www.ibm.com/news/1999/10/11.phtml

[2] http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf

[3] R Saracco, J.R.Harrow and R.Weihmayer "The disappearance of telecommunications" IEEE 2000

[4] M Sloman., "Policy Driven Management for Distributed Systems", Plenum press Journal of Network and Systems Management, Plenum Press

[5] I.W.Marshall and C.M.Roadknight, "Emergent organisation in colonies of simple automata." 6th European Conference on artificial life. 2001

[6] I.Wokoma et al "Weakly consistent sychronisition" Policy 2002

[7] Y.Z. Tsypkin. "Adaptation and learning in automatic systems", Mathematics in Science and Engineering Vol 73, Academic press, 1971.

[8] G. DiCaro and M. Dorigo, "AntNet: Distributed stigmergic control for communications networks", J. Artificial Intelligence Research, 9, pp. 317-365, 1998.

[9] D.A. Fisher and H.F. Lipson, "Emergent algorithms - a new method of enhancing survivability in unbounded systems", Proc 32nd Hawaii international conference on system sciences, IEEE, 1999

[10] M. Gregory, B. White, E.A. Fisch and U.W. Pooch, "Cooperating security managers: A peer based intrusion detection system", IEEE Network, 14, 4, pp.68-78, 1996.

[11] J.H. Holland, "Adaptation in Natural and Artificial Systems" MIT press, 1992.

[12] G. Booth, "GECCO: A Continuous 2D World for Ecological Modeling", Artificial Life, 3, pp. 147-163, Summer 1997.

[13] R. Burkhart, "The Swarm Multi-Agent Simulation System", OOPSLA '94 Workshop on "The Object Engine", 7 September 1994.

[14] S. Forrest and T. Jones, "Modeling Complex Adaptive Systems with Echo", In Complex Systems: Mechanisms of Adaptation, (Ed. R.J. Stonier and X.H. Yu), IOS press pp. 3-21, 1994.

[15] D.H. Ackley and M.L. Littman, "Interactions between learning and evolution". pp. 487-507 in Artificial Life II (ed C.G. Langton, C. Taylor, J.D.Farmer and S. Rasmussen), Addison Wesley, 1993.

[16] I. Harvey, "The Microbial Genetic Algorithm", unpublished work, 1996, available at ftp://ftp.cogs.susx.ac.uk/pub/users/inmanh/Microbe.ps.gz

[17] D.E. Caldwell, G.M. Wolfaardt, D.R. Korber and J.R. Lawrence, "Do Bacterial Communities Transcend Darwinism?" Advances in Microbial Ecology, Vol 15, p.105-191, 1997.

[18] S. Sonea and M. Panisset, "A new bacteriology" Jones and Bartlett, 1983.

[19] N.E. Nawa, T. Furuhashi, T. Hashiyama and Y. Uchikawa, "A Study on the Relevant Fuzzy Rules Using Pseudo-Bacterial Genetic Algorithm" Proc IEEE Int Conf. on evolutionary computation 1997.

[20] I.W. Marshall *et. al.*, "Active management of multiservice networks", Proc. IEEE NOMS2000 pp981-3

[21] I.W.Marshall and C.Roadknight, "Adaptive management of an active services network", British Telecom. Technol. J., 18, 4, pp78-84 Oct 2000

[22] M. Fry and A. Ghosh "Application Layer Active Networking" Computer Networks, 31, 7, pp. 655-667, 1999.

[23] M. Colajanni and P. Yu, "Adaptive TTL schemes for Load Balancing of Distributed Web Servers" Performance Evaluation Review -- ACM SIGMETRICS, 25(2):36--42, September 1997.

[24] I. Chen and C. Chen, "Threshold Based Admission Control Policies for Multimedia Servers", Computer Journal, Oxford University Press, Surrey, GB. Vol 39, no. 9, 1996, pages 757-766.

[25] M. Sipper. "Studying artificial life using a simple, general cellular model." Artificial Life Journal, 2(1), 1995. The MIT Press, Cambridge, MA.

[26] S.Golubic "Stromatolites of Shark Bay" pp103-149 in Environmental evolution ed. Margulis and Olendzenski, MIT press 1999

[27] W.E. Leland, W. Willinger, M.S. Taqqu and D.V. Wilson, "On the self similar nature of ethernet traffic". Proceedings of ACM SIGComm, September 1993.

[28] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, 5, 6, pp 835-846, Dec 1997.

[29] J. A. Shapiro. (1998) Thinking of bacteria as multicellular organisms. Ann Rev. Microbiol. 52, 81-104.

[30] P.Ortoleva, "Geochemical self-organization" OUP 1994