Matching Distributed Systems to their Environment using Dissipative Structures

Jim Dowling, Dominik Dahlem and Jan Sacha
Distributed Systems Group, Trinity College Dublin
Email: jdowling,dahlemd,sachaj@cs.tcd.ie

Abstract

In contrast to a large body of theoretical work on computer systems, distributed systems are not idealised constructions, unconstrained by physical world limitations. They must be designed to account for limiting, real-world properties such as network latency, varying node capabilities, varying application behaviour and unexpected failures. These real-world properties, that we describe under the general area of a system's environment, have regularities or heterogeneities that can often be modelled as a stochastic process, often using well-known distributions. This paper proposes dissipative structures as a model to capture information about properties of these stochastic processes. In dissipative systems, agents (or nodes) sample information from their local environments and collectively build structures that capture knowledge of recent regularities or heterogeneities in the system's environment. Dissipative structures are a promising technique for transferring knowledge of the system's environment among agents without requiring excessive message passing. This approach offers the promise of building more efficient search algorithms based on reduced uncertainty of the system's environment.

1 Introduction

Many existing distributed systems make the assumption that agents (nodes) and their communication links have similar capabilities and resources. For example, Peer-to-Peer (P2P) systems that are based on Distributed Hash Tables (DHTs) often assume that all peers are similar and have equal capabilities for maintaining data, as in Pastry, or assume that the distribution of resources among the peers is uniform, as in Chord [6]. However, these assumptions are known not to be the case in real-life P2P systems, where many properties of peer host environments, such as uptime, average bandwidth and traffic volume, follow a heavy-tailed distribution [7]. For other distributed systems' environments, such as Mobile Ad hoc Networks (MANETs), sensor networks and pervasive computing environments, the properties of their environments may not be as easily captured in a stable, global probability distribution, but may be localised and dynamic. This paper proposes dissipative structures as a general model for capturing the different heterogeneities and regularities in a system's environment.

2 Modelling Distributed System Environments

Many distributed systems, and most P2P systems can be characterised as a system composed of independent, partially connected agents, running on different hosts, that provide a uniform interface to some system service(s). When building large-scale distributed systems that operate in dynamic environments, it becomes difficult for agents to use global knowledge or strict consensus to agree on which agent(s) should be used to solve different sub-parts of the system service(s) or how to find an agent that provides some resource, e.g., the file that we don't know the exact name of in advance. In P2P systems, agents generally take independent decisions on how to find the different agents or resources in the system.

In systems where the cost of all network links and agents are considered equal, an agent has no domain knowledge available with which to direct the search process to find the agent(s) that are most likely to provide the service or resource, i.e., the agents with higher connectivity, uptime, bandwidth and traffic volume. The use of network structures such as a DHT does not help solve this problem, as they provide no domain knowledge of the system's environment to help direct the search problem to better, higher performance agents. The domain knowledge from the system's environment required to help better direct the search process can only be captured by agents interacting with or

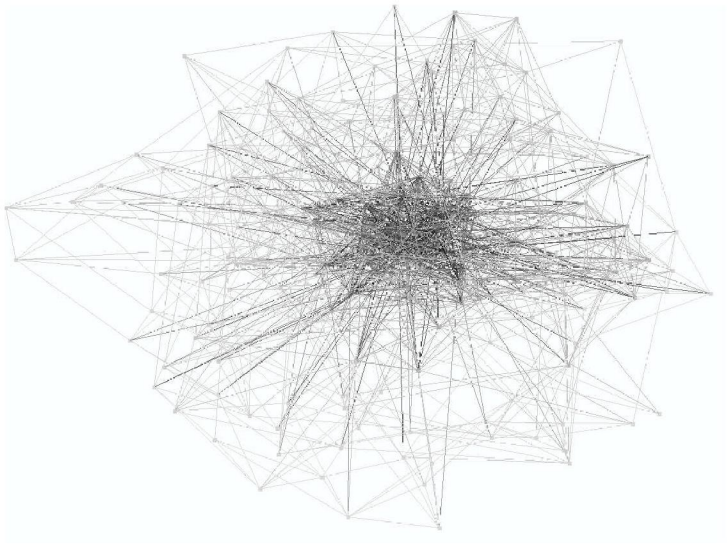
1-4244-0030-9/05/$20.00

sampling properties of their local environments at runtime, e.g., by measuring available bandwidth or available resources, and not at design time.

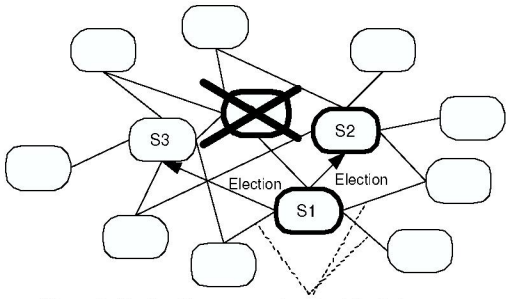
Recently, super-peer architectures have attempted to address this problem by matching high performance peers to super-peers in a two-tier network topology, decomposing the search problem into peer to super-peer and inter-super-peer problems. Peers acquire their domain knowledge of peer properties through users, moderators and sampling their local environments. However, these architectures are binary models, where a peer is classified as either a super peer or an ordinary peer, and they provide, at best, a loose match to the distribution of peer properties found in a P2P system's environment.

In our recent work on this problem, more details of which can be found in [6], we proposed a model where agents interact with their local environment to sample a common set of properties of interest (peer stability, available bandwidth and willingness to contribute resources) and generate a single utility measure from those properties. We then assumed that the distribution of agent utilities in the system is heavy-tailed, as observed in many existing P2P systems [7]. Using this model of an agent's utility, we developed a neighbour selection algorithm where agents preferentially connect to other agents with similar utility levels, and where agents with higher utility levels have higher connectivity. During early experiments on the algorithm, we noted that excessive clustering and partitioning occurs if agents do not constantly re-compute their utility by re-sampling their local environment. We also added long-range random links, and both of these features provided negative feedback on the excessive clustering, producing a variant of a scale-free structure that we call a "gradient" topology [6], (see Figure 1(a)). The topology is called a gradient topology as agent connectivity is more heavy-tailed than in a typical power-law distribution, such as Zipf. Its main characteristic is that over time agents with the highest utilities cluster in the centre of the topology (e.g., agents that would be suitable for providing services), while agents with lower utilities are located at the periphery.

The advantage of the topology is that it captures more fine-grained knowledge of the recent utilities of agents in the system than super-peer architectures. In particular, heuristic search algorithms can use domain knowledge of the system's environment captured in this topology to direct searches towards more highly connected agents in the "core", i.e., towards agents with higher utility that are more likely to provide the services or resources of interest. This heuristic should reduce the amount of messages that need to be sent

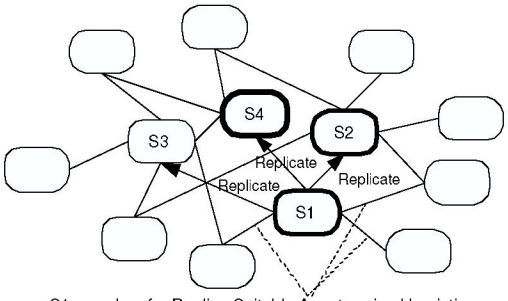


(a) Visualisation of 200-Peer Gradient Topology



S1 sends Election Messages using Heuristic that excludes Lower Connectivity/Utility Agents

(b) The Heuristc enables an Election Algorithm to send a Reduced Number of Messages



S1 searches for Replica-Suitable Agents using Heuristic that excludes Lower Connectivity/Utility Agents

(c) The Heuristc enables a Replica Placement Algorithm to send a Reduced Number of Messages

Figure 1. Heuristic Search Exploiting a Gradient Topology

to solve the problem compared to uninformed search strategies that have no domain knowledge of the system's environment, e.g., many DHT approaches use techniques such as flooding, random walks and expanding ring search.

Two examples of algorithms that we identified that could benefit from using our *gradient topology and heuristic search approach* are master election and replica placement algorithms. The master election problem can be described as a discrete optimisation problem where the problem is to find the "best" agent to act as a new master from the set of agents, (see Figure 1 (b)), while the replica placement algorithm involves finding an agent to host a replica that has a utility level above a pre-defined threshold, (see Figure 1 (c)). Heuristic search algorithms can exploit fine-grained knowledge of agent utilities implicitly stored in the topology to send election or replica messages to only those more highly connected agents. In order to ensure that the domain knowledge captured in our gradient topology is up-to-date, the neighbour selection algorithm runs at discrete time steps, allowing agents to move logically in the topology in order to reflect their current utility.

In summary, in a system where properties of an agent's local environment can be described using a utility value, and the distribution of agent utilities in a system follows a heavy-tailed distribution, knowledge of the distribution can be implicitly captured in a changing network topology that can subsequently exploited by agents to improve the performance of certain distributed systems algorithms.

3 Dissipative Structures

We now introduce dissipative structures, and propose them as a general model of how a stochastic process describing properties of a system's environment can be modelled in a system by groups of agents interacting with their local environments. Study on dissipative structures originated in the field of non-equilibrium thermodynamics, where Ilya Prigogine demonstrated how dissipative structures are ordered structures that arise through an exchange of energy between a system and its environment [5]. Dissipative structures are structures internal to a system that arise when a system is in a *far-from-equilibrium* state, and able to adapt its structure quickly to changes in its environment. Dissipative systems continuously generate entropy internally, i.e., maintain or increase their order, and actively dissipate entropy out of the system into its environment, with the exchange of energy between the system and its external environment keeping them consistent

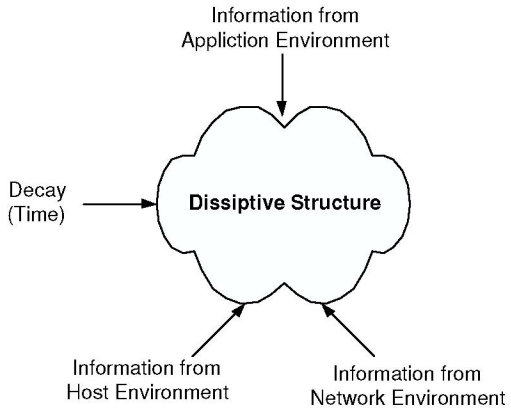


Figure 2. Dissipative Structures in Distributed Systems.

with the 2nd Law of Thermodynamics. A frequently cited example of a dissipative structure is the Bénard cell, where a pattern of hexagonal cells forms on the surface of a liquid through the interaction between externally supplied heat from below and cool air at the liquid's surface [5]. The experiment is repeatable, and shows how self-organising structures inside a system can predictably arise in a given environment.

The gradient topology is an example of a dissipative structure, as it shows how system structure can predictably self-organise through a system's agents interacting with a given environment, i.e., one where there is a heavy-tail distribution of agent utilities. The main difference between dissipative structures in computer systems and those in natural systems is that they require information, not energy, to self-organise. Dissipative structures in computer systems require a constant flow of information between the system and its environment, and they can only exist in conjunction with their environment.

Dissipative structures are built by groups of individual agents executing decentralised algorithms, where each agent continuously acquires information from its local environment, interprets that information, and may then perform local adaptations to its structure, behaviour or environment and communicate any changes to a group of neighbouring agents (typically a small number of agents relative to the system size). The adaptation by an agent of its behaviour, structure or environment is the mechanism through which information acquired from its local environment is encoded in the dissipative structure. Inter-agent communication is necessary to enable agents to provide feedback to their neighbours about changes in the environment. Feedback between agents, whether direct or stigmer-

gic, enables rapid, cascading changes to a dissipative structure, allowing the structure to quickly adapt to changes in the environment. In the absence of information input from the agents' environments, dissipative structures should decay their structure to prevent the information they contain becoming stale.

From a more formal perspective, dissipative structures are multi-agent memory models that are constructed using information from the local environments of groups of agents. The environment of a group of agents can be modelled as a stochastic process, and a dissipative structure is a self-organised representation of that stochastic process that captures information about the process, e.g., it captures regularities or heterogeneities in the group's environment. Dissipative structures are typically localised or global macro-level patterns in a system that can be observed (allowing information extraction) by entities external to the system. Many different self-organised representations of dissipative structures are possible. For example, the gradient topology is a global self-organising pattern that contains information about agents' environments, but other localised self-organising patterns are also possible.

An important point, however, is that the macro-level patterns representing the information in dissipative structures is not directly accessible by its constituent agents as they only have a partial view of the system. However, a goal when building dissipative systems is to enable agents to exploit the knowledge captured in dissipative structures, e.g., to build better searching algorithms. This is a challenging problem, although we have shown one example of how this is possible by designing heuristic search algorithms that exploit domain knowledge captured in the gradient topology. However, the more general problem of how to design decentralised algorithms that can exploit the knowledge in a dissipative structure but do not have advance knowledge of its representation is still an open problem.

Other examples of dissipative structures can be found in existing distributed systems. These include converged trails in Ant Colony Optimisation systems that are constructed through ants interacting with their local environments [2], and in the SAMPLE routing protocol [3] where stable paths to popular routing destinations emerge by routing agents sampling the quality of network links in their local environment and collectively learning the more optimal routing paths. In these systems, groups of agents with converged routing tables are dissipative structures that require a constant flow of information (from ants or routing packets) to maintain their structure or else they decay. The dissipative structures have a different representation in these systems than in the gradient topology. As a pattern, the converged routing tables (i.e., the dissipative structures) can be localised in the system, and they are formed by agents locally adapting the strength of their connections to their neighbours (i.e., the values of their routing table entries). In contrast, the gradient topology is formed by agents reconfiguring their connections to their neighbours. However, the basic principle of agents requiring information from their local environments and performing local adaptations to build self-organising structures that capture information about those environments is common to these different systems. A schematic model of the basic requirements for dissipative structures to form in distributed systems is illustrated in Figure 2.

Finally, dissipative structures contrast with other decentralised approaches to information capture in distributed systems, including aggregations and population protocols [1], as they are represented as sub-symbolic patterns, decay in the absence of information flows and can model localised, as well as global, stochastic processes in a system's environment.

4 Research Agenda for Dissipative Systems

We see an important use for dissipative structures in helping to build distributed systems that better match properties of their stochastic environment. In particular, the research areas we are addressing include:

Environments Investigate different distributed systems environments with a view to describing localised or global properties of their environments using different stochastic processes.

Structures Scale-Free and Gradient topologies are just two of many different possible network structures. It has been shown that the distribution of local structures, i.e., network motifs [4] or sub-graph patterns, carry significant information about the large-scale topological organisation. We are investigating these network motifs as a possible approach of how to build dissipative structures where agents can specialise and organise into a functional unit (i.e., a motif).

Specialisation Information captured in dissipative structures can potentially be used to enable agents specialise to provide different system services. Agents could use information from the dissipative structure, e.g., their relative position in the structure, to decide either individually or as groups how to specialise, and positive feedback mechanisms

can enable the recruitment of agents to a group. Negative feedback can be designed to induce the formation of boundaries between groups. If specialised groups of agents have formed, other agents can then discover and use their services using heuristic search algorithms that should be adaptable to failed agents and communication links. Self-organised speciailsed groups should also be able to adapt their memebership to changes in their application environment, e.g., by recruiting new agents in response to increased service usage.

Many Multiple dissipative structures can allow us to model different properties in a system's environment, e.g., agent utility and agent reputation. If the structures represent orthogonal stochastic processes in a system's environment, this may be feasible, but if they are not the algorithms that generate the different dissipative structures may have unintended effects on one another.

Learning For dynamic environments that can be described by different stochastic processes over time or location, there is a challenge of how groups of agents can collectively adapt dissipative structures to better capture information about the current state of their environment. There is also the challenge of how individual agents can learn to exploit the information captured in a dissipative structure.

Memory In dissipative systems, groups of agents generally store some collective memory model of their environment. At the individual agent level, tradeoffs generally need to be found as to how individual agents build and decay their local part of memory model so that it accurately reflects the current state of the environment.

Experimentation Bifurcation analysis has been used to study properties of dissipative structures in controlled environments. There is wide scope for applying bifurcation analysis as well as developing new experiment methodologies to demonstrate the stability characteristics of dissipative structures in different environments.

Trust If we wish to deploy our Gradient topology in an open distributed system we require a trust model for inter-agent interactions. This is a general problem that applies to any open multi-agent distributed system that aims to use dissipative structures.

5 Conclusions

In contrast to idealised network structures commonly found in distributed systems, such as DHTs, dissipative structures are embodied structures that are constructed at runtime using information acquired from the system's environment to produce a structure that contains knowledge of the state of the system's environment. We have shown how a P2P system, where the utility of agents' local environments follows a heavy-tailed distribution, can become globally ordered, in a predictable manner, through the system's agents interacting with their local environment and preferentially attaching to one another. Our research plan is to further investigate stochastic processes that describe different distributed systems environments, and the types of dissipative structures that can be built to capture information about these environments. Our hope is that dissipative structures will enable future distributed systems to more closely match properties of their environments, and that we will make progress towards finding system structures that benefit system function.

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 290–299, New York, NY, USA, 2004. ACM Press.

[2] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In *New Ideas in Optimization*, pages 11–32. McGraw-Hill, London, 1999.

[3] J. Dowling, E. Curran, R. Cunningham, and V. Cahill. Using feedback in collaborative reinforcement learning to adapt and optimise decentralised distributed systems. *IEEE Transactions on Systems, Man and Cybernetics (Part A), Special Issue on Engineering Self-Orangized Distributed Systems*, 35(3):360–372, 2005.

[4] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002.

[5] I. Prigogine and I. Stengers. *Order Out of Chaos*. Bantam, 1984.

[6] J. Sacha and J. Dowling. A self-organising topology for master-slave replication in p2p environments. In *the 3rd International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, pages 51–63, 2005.

[7] S. Sen and J. Wong. Analyzing peer-to-peer traffic across large networks. In *IEEE/ACM Transactions on Networking*, volume 12, pages 219–232. ACM Press, April 2004.