



BISON IST-2001-38923

*Biology-Inspired techniques for
Self Organization in dynamic Networks*

Structure/Function CAS Matrix

Deliverable Number: D03
Delivery Date: January 2004
Classification: Public
Contact Authors: Geoffrey Canright, Andreas Deutsch, Niloy Ganguly, Márk Jelasity
Document Version: Final (January 30, 2004)

Contract Start Date: 1 January 2003
Duration: 36 months
Project Coordinator: Università di Bologna (Italy)
Partners: Telenor Communication AS (Norway),
Technische Universität Dresden (Germany),
IDSIA (Switzerland),
Santa Fe Institute (USA)

**Project funded by the
European Commission under the
Information Society Technologies
Programme of the 5th Framework
(1998-2002)**



Abstract

We present a tabular way of organizing ideas and results from studies of complex adaptive systems (CAS), operating on some network structure to implement some desired function. The term “matrix” is then used loosely: the tabular structure which we present has three dimensions, ranging over the function implemented, the network structure (here stated more precisely in terms of topology), and the CAS (here analyzed in terms of a small set of microscopic agent mechanisms). For each triplet (function, topology, CAS), we would like to collect information about the performance achievable. For example: do ant-like mechanisms perform routing well on a scale-free topology? Answers to such questions will be collected in the course of work in the BISON project. Here we present the “matrix” structure itself, ie, the nature of its three dimensions, and give some illustrative examples of how it might be filled in.

Contents

1	Introduction	4
2	Topology replaces structure	5
3	Microscopic mechanisms	7
4	Functions from microscopic mechanisms	8
4.1	Routing	8
4.2	Search	10
4.3	Aggregation	11
4.4	Topology Management	13
5	Outlook	13

1 Introduction

The BISON project seeks to systematize the application of biology-inspired, complex adaptive systems to practical technological problems. As one aspect of this effort towards systematization, we have envisioned a three-dimensional matrix, of the following form: one dimension ranges over a set of network *structures*; another ranges over *functions* to be accomplished on the network; and the third dimension ranges over the *biological systems* which may be used to accomplish the function on the network structure. The matrix entries are then some measure of the success or effectiveness of biological system BS in accomplishing function F on network structure S.

This document builds on the rest of the work accomplished during the first year of BISON. The concepts that we refer to here are discussed in more detail in other deliverables: issues related to topologies and functions in deliverable D01, the classification and explanation of microscopic mechanisms in deliverable D02, and the protocols that are classified here can be found in deliverables D05 and D08.

It is of course out of the question that the BISON project might obtain meaningful results for all possible entries in this matrix. Instead, by the end of the BISON project, we expect to have a significant number of matrix entries—enough that one might be able to begin to extract some understanding of why some ideas work well, while others do not.

At present, very few of the matrix entries are known. One can however place *guesses* in the matrix. Such guesses serve to organize future research, and the matrix in this sense serves another goal, namely that of stimulating and collecting ideas for research.

We can represent on paper any two-dimensional slice of the three-dimensional object that we call the structure/function “matrix”. An early visualization showed structures and functions, with the CAS-dimension implicit. Here we propose to replace the word “structure” with the more precise term “topology”. As examples of “structures” we have given ad-hoc networks, peer-to-peer networks, etc. These terms are imprecise, and often have an implicit or explicit notion of function, entangled with a more or less ill-defined topology. Hence we focus on *topology* as a more precise description of network structure. In Section 2 we will discuss in more detail the use of topology for the “structure” dimension of the matrix.

In deliverable D02 we began a process of analyzing our various CAS of interest in terms of the *microscopic mechanisms* which give rise to their collective behavior. Specifically, we presented a short list of microscopic mechanisms, and analyzed four idealized biological systems—ants, immune cells, amoebae, neurons and viruses—in terms of this short list. This analysis, again, takes the form of a matrix: (idealized) biological system vs. microscopic mechanism, with the matrix entries being Yes or No. In Section 3, we repeat and update this matrix, in order to make the present document more self-contained.

In Section 4 we present yet another matrix, by arraying *functions* against microscopic mechanisms. We claim that this new matrix is a useful intermediate step in developing guesses for the structure/function/CAS matrix entries. The reasoning is straightforward: we think of a (bio-inspired) CAS as a set of microscopic mechanisms (Section 3). In Section 4, we analyze functions (rather than biological systems) in terms of microscopic mechanisms. The point of view is that it is the microscopic mechanisms which give rise to the collective behavior (function).

The biological systems are just, in this viewpoint, collections of microscopic mechanisms—collections whose collective behavior is known. Ultimately, we aim to wean ourselves of the dependence on biology, through identifying a set of microscopic mechanisms which we can freely combine to build new CAS, independently of their existence in biology.

Hence the third dimension of our “matrix”—ranging over biological systems—might be better taken as ranging over all possible sets of microscopic mechanisms. This (plus the replacement of “structure” with “topology”) rephrases the question which each matrix entry tries to answer: instead of asking what is the effectiveness of biological system BS in accomplishing function F on network structure S, we now ask what is the effectiveness of set {MM} of microscopic mechanisms in accomplishing function F on topology T. This new form of the question is more precisely posed than the old form. However, it is not practical to investigate or even list all possible sets of microscopic mechanisms. Instead we will look at a few combinations in Section 4—some inspired by a biological system, others not.

Finally, we offer a summary and outlook in Section 5.

2 Topology replaces structure

In Deliverable D01 we classified structures into two types: overlay networks and ad-hoc networks (MANETs). The former is a logical network, overlaid on a physical network infrastructure — typically the Internet. Overlay networks are most commonly employed when routing and connectivity is cheap and easy, so that logical connections may be built without severe constraints from the physical infrastructure. For MANETs, in contrast, connectivity and routing are problematic. Hence the physical infrastructure becomes the object of interest itself, rather than being a dependable and rather invisible background on which other structures are built.

This distinction—basically physical vs logical—is useful and meaningful. However it does not address directly what type of topology or connectivity a given structure has. It is a primary purpose of BISON to find means to ensure good performance on networks. Two of the most dominant and general elements determining performance are topology and algorithm. It is of course an explicit premise of BISON that algorithm (mechanism, CAS) has an important effect on performance. Given however the great variation in possible types of topologies, and in their properties such as path length, mixing, clustering, etc, there is also little doubt that topology has large effects also on performance.

Therefore we replace the structure dimension of our matrix with topology. A representative set of topologies was presented and discussed in Deliverable D01, and so will not be repeated here. Our point is simply to underscore that, in seeking to understand the main determinants of performance of a function, one must look at topology as well as algorithms.

Our two main structural types—overlay networks, and MANETs—must then be represented in terms of topology. Here the differences between the two are large. Overlay networks can take on any of a wide variety of types of topology—again, see D01. MANETs, in contrast, are typically confined to a single type of topology, which may be termed a ‘spatial network’ [5]. The determining characteristic of a spatial network is the existence of a distance scale for links, and a cutoff in the spatial range of the links. In short, one can say that there are no long range links in such a topology. This succinct statement captures the constraint of such a topology;

but it leaves implicit the fact the the topology even *allows* the definition of a length scale (other topologies, such as random or scale-free, do not).

We note further that the (common) terminology that we use for topologies makes no explicit reference to any dynamic nature of the nodes or links. Of course, BISON will study dynamic topologies. Typically, for overlay networks, one represents the dynamic nature in terms of uncorrelated, probabilistic loss and gain of nodes and/or links. For MANETs, a great deal of work has been done with more detailed models for mobility and connectivity of the nodes. These more detailed models can also be reduced to simple probabilistic models. In any case—regardless of the model for dynamic availability of nodes and links—these changes occur with respect to a ‘reference’ topology. The *reference topology* is typically specified statistically; hence there is not a great loss of information in using the static (but statistical) terminology for the various types of topology. Furthermore, in defining a reference topology, we must assume that any node/link dynamics do not have the effect of *changing* the reference topology. Some topologies (such as scale-free) maintain themselves naturally, while others must be actively maintained by topology management. In order to useful results about how to implement functions on a network, we confine ourselves to those situations where the dynamics are not so severe as to make it impossible to define a reference topology. In short: our terminology for topologies defines a reference topology, with the implicit addition of statistical fluctuations in detail, but persistence of the reference topology.

Hence our ‘matrix’ dimensions range over (function), (biological system or set of microscopic mechanisms), and (reference topology). One possible 2D slice of this matrix shows reference topology, and function. Such a slice is illustrated schematically in the table below.

	Lattice	Tree	Random	Small world	Spatial
Search					
Routing	-	+	-	depends	-
Monitoring					
Aggregation	local	depends	+	+	local
Distributed processing					

Table 1: Matrix of functions vs topologies. The dependence of the performance of the functions on topology is rendered visible in this form, while the dependence on mechanism (eg, CAS) is suppressed.

We do not attempt to fill in all the entries in this slice, as too many are unknown. Instead, we focus on two rows of Table 1—routing, and aggregation—and give some comments on the entries given. These two rows are given for illustrative purposes. We expect that, at the conclusion of the BISON project, we will have much more to say about the entries in our “matrix”.

Routing on networks is one of the better-studied functions; and so we have marked that row of the table. The tree topology gets a + as it allows for efficient routing; while lattices and random graphs get – as they do not. Small-worlds graphs may or may not—they are defined by two properties (path length and clustering), and these two properties allow for some topologies that do not allow for the efficient finding of paths [2, 3], and others that can.

Aggregation (see Section 4.3) shows a rather colorful picture. For a (low dimensional) lattice topology and for a spatial topology (defined as a topology in which spatially “close” nodes are connected, like in an ad hoc network, and so there are no long range links) aggregation can be efficient locally but not globally because the average shortest path length in these topologies—which is a lower bound of the time necessary for any influence to reach a node—is high. On a tree, one can implement a special kind of aggregation in which the root node can collect and aggregate information from the network, however in this case only the root will learn the aggregate value. On a random topology it is possible to define an efficient aggregation protocol as we show in deliverable D05, where all nodes calculate the aggregate simultaneously. In many senses any small world topology can be efficient also, as the main requirement is to have a low average shortest path length. However, in some small world topologies, like for example in a scale free topology, there might be communicational bottleneck nodes which reduce the efficiency of aggregation.

This table may give the impression that topology is the sole determinant of performance. That of course is not true. What is true is that topology can place bounds on what kind of performance can be achieved. Any + and – entries in the above table then imply that efficient algorithms can or cannot, respectively, be found for the given topology. The algorithms themselves (ie, the CAS) lie in the unseen, third dimension of the table.

3 Microscopic mechanisms

Here we repeat a table from Deliverable D02, which analyzes some biological systems in terms of a small set of basic, microscopic mechanisms.

	Ants	Immune cells	Amoebae	Neurons	Viruses
Memory (1)	x	x	x	x	-
Selection (2)	-	x	-	-	x
Proliferation (3)	-	x	-	-	x
Response/interaction (4)	x	x	x	x	-
Signaling - diffusive (5a)	-	x	x	-	-
Signaling - nondiffusive (5b)	x	-	-	x	-
Mobility (6)	x	x	x	-	x
Mutation (7)	-	x	-	-	?

Table 2: Five idealized biological CAS, analyzed in terms of a small set of microscopic mechanisms. The collective behaviors of these idealized CAS are built from the microscopic mechanisms shown.

In D02 we remarked that the chosen microscopic behaviors in a given column (eg, Ants) may plausibly give rise to the collective behaviors (finding paths to food) observed in that biological system. In the next section we will try to present this “plausible dependence” of collective behavior on microscopic behaviors more explicitly. The table above, from D02, is then reproduced

here solely for convenience of reference.

4 Functions from microscopic mechanisms

Here we present our standard set of basic functions, and for each function, one or more sets of microscopic mechanisms which may plausibly or provably give rise to the function in question. We limit ourselves to basic functions, as the advanced functions are best understood as a higher level of organization, ie, as composed of basic functions.

Also, we omit two basic functions from our list, for differing reasons. We have dropped the discovery function from our list, as it presents too little technical challenge, and is too close to the search function in nature (see Deliverable D01). The monitoring function, as pointed out in D01, is an example of collective computation (also called aggregation). Monitoring is so important to network operation that we will retain it in our list. However, in the table below, we omit monitoring—since the ability to carry out aggregation implies the ability to perform the monitoring function. We are then left with four basic functions:

	Routing	Search	Aggregation	Topology Management
Memory (1)	x x x	x x x	x x x	x
Selection (2)				
Proliferation (3)	x		x	
Response/interaction (4)	x x x	x x x	x x x	
Signaling - diffusive (5a)			x x	
Signaling - nondiffusive (5b)		x	x	
Mobility (6)	x x x	x x x	x x	x
Mutation (7)			x	

Table 3: Four basic functions, analyzed in terms of the microscopic mechanisms which may be used to accomplish the function. For some functions—Routing, Search, and Aggregation—three candidate sets of microscopic mechanisms are given. The example sets given are not exhaustive. One such set may be viewed as “minimal”—using the fewest microscopic mechanisms—while the others are “embellished”. Embellished sets are of interest since the minimal set may give poor performance.

Now we discuss the functions in turn.

4.1 Routing

Routing is the task of finding a path—generally, a ‘good’ path, according to some goodness criteria—from a sender S to a desired destination D . The protocol may be dependent on global topological information, as is the link-state protocol [4]; or it may be some sort of incremen-

tal/stochastic protocol where each node looks at the next node and decides the next hop. An example of the latter is the Distance Vector Routing Protocol [1]. Obviously, we are only interested in protocols using local information. However, the distinction is not totally sharp, as global information may be obtained from local information using the basic function termed aggregation or collective computation.

There are several different possible criteria for a ‘good’ path. However, in this discussion, we just concentrate on exploring the set of microscopic (agent) behaviors which can, arguably or provably, give rise to the function at all—without regard to goodness of the performance. For this purpose, we consider three different strategies which can be utilized. This is of course not an exhaustive set. Our aim is rather to find some *minimal* algorithms which will accomplish the function, and also some more ‘embellished’ algorithms, where the embellishments might give better performance.

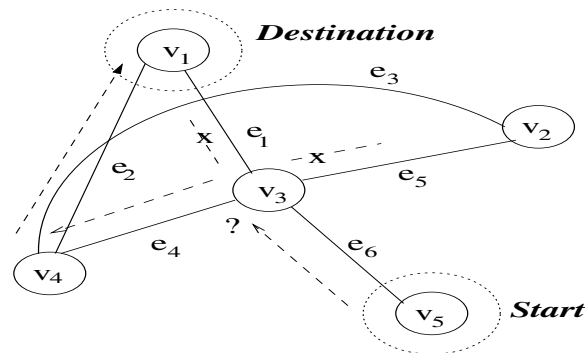


Figure 1: Example of Routing in p2p network. In this example, the request for a path is initiated by the source node v_5 ; and the destination node is v_1 .

Flooding Flooding is illustrated in Fig. 1. In flooding, packets are emitted from v_5 . The packets contain information regarding the initial node (v_5) and the final destination (v_1). On reaching v_3 (in the example, v_5 just has one outlink) the packet duplicates, and moves in all possible directions (v_4 , v_2 , and v_1). When one packet reaches v_1 , its state is changed, indicating that the packet has arrived at its destination. When other packets reach the destination, they are simply discarded. So from this small example, we see that our agents, in order to do routing by flooding, must have state (memory) in order to remember both the destination D and the path taken. They must be able to proliferate in order to flood; and they must be mobile. Finally, they must be able to change state in recognition of the destination D . Hence we get the first column under Routing.

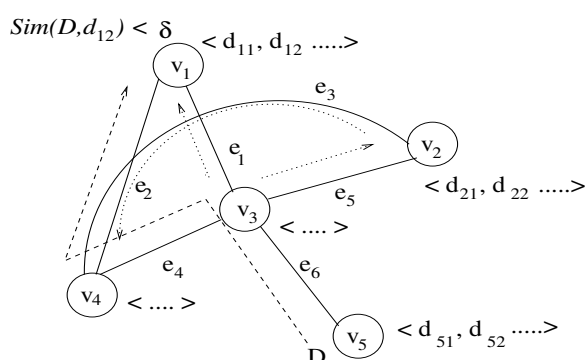
Random Walk We can also accomplish routing with a random walk. The agents also in this case will possess the information regarding the initial path (v_5) and the final destination (v_1). Packet(s) are still emitted from v_5 . On reaching v_3 , the packet, unlike flooding does not duplicate, but simply take a randomly chosen path (say v_4), and ultimately establishes contact between v_5 and v_1 via $v_5 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_1$. Here we see that the agents need all the features of the flooding mechanism (memory, response, and mobility), except proliferation.

Ants Finally we mention ants. Ants employ the same mechanisms as a random walk: memory, response, and mobility. We refer again to the figure, now for the case of an ant algorithm.

The packet (ant) which is emitted from v_5 , on reaching v_3 , decides the next hop on the basis of certain *information*. This is presumably information about route quality or goodness, such as hop distance to the destination, or network congestion in that route. More generally, we view this information as a nondiffusive signal (pheromone) left by earlier ants. Therefore, in this case if the path establishing contact between v_5 and v_1 is $v_5 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_1$, the path is a result of some collective memory of the system. This collective memory arises from the use of a nondiffusive signal. Hence the third column under Routing.

4.2 Search

Before listing any microscopic behaviors, we first briefly discuss the basic search mechanism and some different ways a search function can be defined. Let the network (G) be denoted as $G = (V, E)$, where each node v_i holds components (data/files) $d_{i1}, d_{i2} \dots$. A particular node $v_i \in V$, wants to perform a search of some keyword or category, or tries to find a particular file (say) D . That implies that one or more query message packets, carrying the search criteria, start from v_i and travel in the network. The packets look for some match to the query in the network. The match may be exact ($D = d_{ij}$) or it may be within some threshold distance $\text{Sim}(D, d_i) < \delta$, where δ implies some threshold distance. The search may demand for a single answer, or a number of hits.



The figure shows search strategy followed by a single packet performing a single search. However, there can be several packets each searching for multiple hits. In the figure, search is initiated by the node v_5 . The searched item is D . A query message packet travels from $v_5 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_1$. When it reaches v_1 , in node v_1 , D matches with d_{12} and a search result is obtained.

Figure 2: Example of search in p2p network

Search is similar to routing in many aspects: routing is, after all, a search for a destination, with the added assignment of remembering and evaluating paths to that destination. If we drop the requirement for path, and widen the nature of the thing searched for, then we get the search function. The memory requirement is reduced, as information about paths is no longer required. But it is clearly not zero: the searching agents must retain memory of the target, and perhaps some goodness-of-fit criteria.

For example, in *Figure 2*, search is initiated by the node v_5 . The searched item is D . A query message packet travels from $v_5 \Rightarrow v_3 \Rightarrow v_4 \Rightarrow v_1$. The movement can either result from a *random walk*, or from *flooding*—as was the case with routing packets. The packets carry with them information about the target D , and the threshold level (δ), defining how much deviation from D is allowed to ensure a match. Return movement is triggered when an agent finds a matched item—since this information must be carried back to v_5 . Therefore, mobility is needed, both to enable the search, as well as the response when the target is recognized. Hence, for search

using a random walk, we get the same basic three microscopic mechanisms: mobility, memory, response. (See the first column under Search.) If we use a flooding scheme (not shown in Table 3), then the proliferation mechanism is also needed.

We can also use the ants' nondiffusive signalling (second column under Search). Note that, in the case of routing with ants on a nonhierarchical topology, there must be one type of signal (pheromone) for each possible destination D ; whereas, in this case (search), it is impractical to have one type of signal for each possible search. Instead there must be some coarse-graining. For example, one could have one pheromone species for each keyword. In spite of interesting differences such as this one, we see in fact that all three sets of microscopic mechanisms which were used for routing in the previous section can also be used here for search.

As a final example we mention the immune-system mechanism for search, as described in Deliverable D08. Here the immune cells supplement the basic three mechanisms with two more: proliferation and mutation (third column under Search). Unlike in the case of flooding, however, here the proliferation is not blind, but rather determined by local information—the quality of the match found. For example, in *Figure 2*, the proliferation at the node (say) v_3 will depend upon the state—that is, quality of the match—found at v_3 . Therefore, here the mechanism 'response' actually involves more complex behavior (both recognition *and* the triggering of proliferation) than in the previous cases.

Mutation is added, inspired by the immune system, to allow for a broadening of the search. That is, the target D may be modified to D' , again based on local information such as quality of match. The basic idea is that both proliferation and mutation can be triggered by a good enough match, so that the neighborhood of the good enough match may be explored thoroughly, exploiting correlations between content and network neighborhood (see D08 for a more detailed discussion).

Note that a simple *coupling* between microscopic mechanisms—such as, in this case, recognition and proliferation—can give rise to quite different behavior from the uncoupled case (flooding). (Again see D08 for details.) The point to be made here is then that identical columns in Table 3 can give rise to distinct behaviors. Such differences in behavior are to be expected, since the state spaces and transitions implied by the memory and response mechanisms can vary enormously—with the kind of coupling mentioned here being one example.

4.3 Aggregation

Aggregation, or collective computation, entails a collective form of information processing by all the nodes, in a way that uses little individual processing by each node. In other words, the processing is done by message passing. Hence we need the two basic mechanisms of message passing: mobility and memory. These messages must however have *some* interaction at the nodes; otherwise nothing happens. Hence we again need response (interaction). This minimal set of three is the same as that found for routing and for search.

An example clarifies the above concept. Let the network be represented by a graph $G = (V, E)$, where a node $v_i \in V$ wants to enquire about some nonlocal or even global parameter. (An example enquiry related to a global parameter may be when the node wants to know the collective capacity of storage in the network). The enquiring node sends some *aggregation packet* which triggers each node to compute the requisite information in its local node. Then the in-

formation is passed to the target node v_5 , which aggregates the result (Fig. 3(a)).

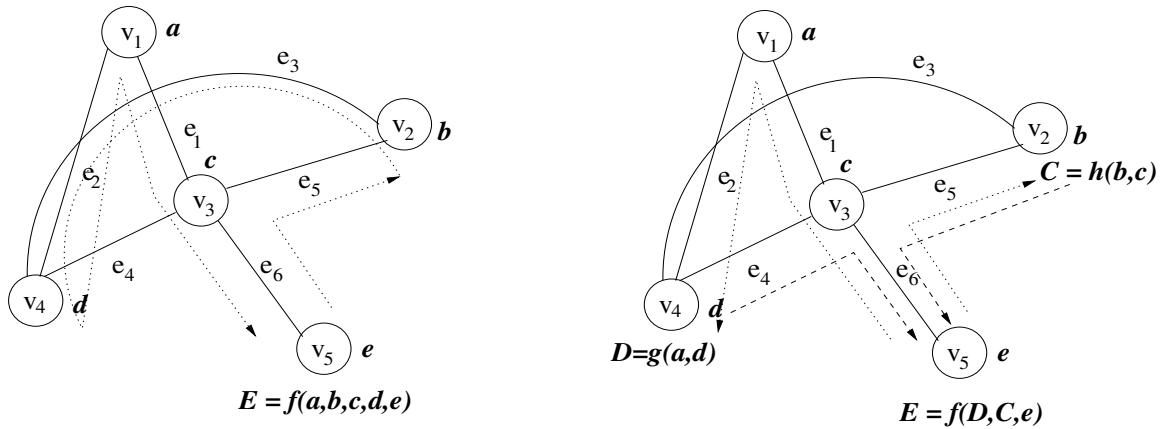


Figure 3: Aggregation in a network

This is a primitive form of aggregation in which most of the computational work is done by the enquiring node. Fig. 3(b) represents a different approach. In this case, nodes at some distance from the enquiring node can perform some partial preprocessing, sending their partially-aggregated results further. In the Figure, nodes v_4 and v_2 compute $D = g(a, d)$ and $C = h(b, c)$, respectively; and then the result is passed to the target node v_5 . Clearly, there is an interplay between routing of the enquiry, and the computation and routing of the partial results. This interplay can be simple or complex, depending on the function to be computed collectively. Yet, as long as message passing is the sole means employed, we find the same three basic microscopic mechanisms at play.

Now we consider the amoebae in a slime mold. These cells perform a collective computation in which the entire colony participates, and the quantity computed is (roughly) the average degree of hunger of the colony. It is interesting to note that, if the colony is hungry enough, it is triggered to *physically* aggregate, ie, to form a multicelled body which can move around, and spread spores. The amoebae use the “basic three” microscopic mechanisms. That is, they are mobile, have state, and change state in response to their environment. In addition, they emit a diffusive (mobile) signal (cAMP) which plays a vital role in coordinating the collective estimation of the colony’s degree of hunger. Hence we get the middle column under Aggregation. In deliverable D05 we describe our protocols to aggregation which are closest to this third approach.

Finally, we find it plausible that the mobility of the amoebae themselves could be dispensed with in performing the collective computation. It is of course necessary for the *physical* aggregation — but that is another function. The point is that the diffusive signal, plausibly, may enable the kind of self-amplifying response to hunger that gives rise to the computation, without employing any motion of the emitting and receiving cells. Hence the third column lacks agent mobility. It is in fact the *only* column that lacks mobility. However, since signals are in fact just primitive agents with memory (stability) and mobility (diffusion), the mobility has simply shown up in another form. Nevertheless it is of some interest to consider whether diffusive signalling could be used in the other two basic functions considered here. They are used in an

advanced function (see D08) to accomplish load balancing: here the signals and the load itself are the only mobile entities.

4.4 Topology Management

The complexity of a topology management protocol can be anything from extremely complex to extremely simple. Furthermore, for all different topologies, the requirements may differ. For illustration, we discuss the protocol we suggest in deliverable D05.

This protocol attempts to maintain a random overlay topology in the face of node dynamism and failures. The approach that is taken is relatively simple. Each node maintains a set of addresses, which define the neighbors of the node. Periodically, the nodes exchange these sets with a random neighbor, and update their own set based on the received set.

This mechanism can be characterized by simple message passing in terms of microscopic behavior. In our framework, a message can be thought of as an agent which has *memory* (remembers the information it has to carry) and which is *mobile* (it can transfer information from one node to another). Hence the resulting set of microscopic mechanisms is extremely simple (simpler even than our model of a virus!).

5 Outlook

It is a highly nontrivial task to systematize knowledge about the behavior of complex systems. This deliverable, and the “matrix” that it describes, present, not systematic results, but only a program for doing such systematization. The program requires well over three years to complete. However, we believe that the BISON project can make a significant contribution to such systematization in the course of three years. This document presents some early efforts in this direction.

We have identified topology and CAS as two of the dimensions of our matrix—because we believe they are the two primary determinants of performance of functions on networks. Our matrix thus poses some very interesting questions, without yet answering them: what is the relative importance of these two elements in determining performance? will the answer vary according to the function? Might there be cases where one or the other is completely dominant?

Further questions are posed by the matrix in Section 4. We note that this table purports to predict collective behavior from microscopic behavior. In some cases, this kind of prediction is easy. In others, it is assumed to depend upon “emergence”, meaning, perhaps, that prediction of the global collective behavior is *not* easy. One very interesting question is then, to what degree is such prediction possible? To how many distinct functions, and distinct kinds of functions, can it be extended? To what extent can it be moved from intuition, guessing, and simple logic, to a more coherent theory?

These questions have both scientific and technological interest. This document presents a framework for working towards answers to these questions. The BISON project will test a number of the models described in Section 4, and the results will yield partial answers.

References

- [1] Christian Huitema. *Routing in the Internet*. Prentice-Hall, Inc., 1995.
- [2] Jon Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.
- [3] Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.
- [4] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, London, 1996.
- [5] Duncan Watts. *Small Worlds*. Princeton University Press, 1993.