BISON
IST-2001-38923

Biology-Inspired techniques for Self Organization in dynamic Networks

Models for Basic Services in AHN, P2P Networks

Deliverable Number:	D05
Delivery Date:	January 2004
Classification:	Public
Contact Authors:	Gianni Di Caro, Frederick Ducatelle, Niloy Ganguly, Poul Heegaard, Mark Jelasity, Roberto Montemanni
Document Version:	Final (February 2, 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 5^{th} Framework (1998-2002)

Abstract

This document reports on the models developed so far for basic functions in dynamic networks. The basic functions of interest in BISON, as previously defined in deliverable D01, are: routing, search, topology management, collective computations and monitoring. For each of these basic functions one or more models have been developed, where each model has been designed to match the characteristics of either overlay networks or mobile ad hoc networks. Models are described by discussing their core characteristics and pointing out analogies with biological systems and relationships with related work. In particular, the presented models have been designed utilizing notions from ant colonies (for routing and monitoring in mobile ad hoc networks), immune system (for search in P2P networks) and epidemics (for topology management, collective computations and monitoring in overlay networks). For some models some preliminary results were already available and have been shortly discussed. However, more precise and in-depth results will be reported on deliverables D06 and D07 due by month 24.

Contents

Introduction

This deliverable describes the models that have been developed so far in BISON for the management of basic functions in dynamic networks. The content of this deliverable is complemented by that of deliverable D08, which deals with advanced functions. The basic functions studied in BISON are discussed in deliverable D01. They are the following ones:

- routing,
- search,
- topology management,
- collective computations,
- monitoring.

In this deliverable each set of models developed for each different basic function is discussed in a separate part of the document. According to the fact that mobile ad hoc networks and P2P / overlay networks are seen in BISON as networks possessing rather different core characteristics (e.g., see D01), each part can have different sections each discussing a different model specifically designed to deal with either overlay or mobile ad hoc networks.

Each model description is organized such that, in addition to the description of the characteristics of the model and its context of application, the reader can find discussions on: (i) biological analogies and/or inspiration behind the model, (ii) relevant related work, (iii) preliminary experimental results, if already available at this stage of the project.

More specifically, this document describes the following models / algorithms:

Routing: Section 1 discusses three algorithms, inspired by the pheromone-driven shortest path behavior observed in *ant colonies* [66, 35, 46], for adaptive routing in mobile ad hoc networks (MANETs). The three algorithms constitute a hierarchy of mobile-agents-based algorithms aimed at covering a wide spectrum of possible scenarios in terms of network types, sizes, connectivity, mobility and traffic patterns.

Routing is one of the most critical functions in MANETs. Only after routing is delivered in a robust and efficient way, it is possible to start considering other basic or advanced services. According to this fact, so far our research on MANETs has mainly focused on routing and monitoring (see Section 7), which we see as a function that can be directly built on top of our adaptive ant-inspired routing protocols.

Search: In Section 2 of this deliverable we describe an approach to search in overlay networks that does not assume organized structure and relies only on intelligent random walks that "proliferate" and "mutate" in a tree-like fashion. The proposed algorithm has been directly inspired by concepts from the study of the *immune system*, a highly parallel and distributed adaptive system which deals in effective way with problems of recognition and classification (of external and internal pathogens). In the model the search query is conceived as an antibody which is generated by the node initiating a search whereas the

antigens are the searched items possibly hosted by other members of the P2P networks. Similarly to what happens in the immune system, search queries undergo proliferation, activation, induction, differentiation, attack and memory formations. Moreover, the overlay topology of the network is updated according to search results, such that the obtained topology clusters together nodes hosting similar items.

Topology management: Communication topology plays a key role in defining the constraints on any function implemented on a given network. In the case of overlay networks there is a large freedom of building arbitrary topologies so topology can be optimized to support functions like routing, search or load balancing. Even in ad hoc networks, one can change the topology by changing the range of communication of each node thereby minimizing collisions and saving power while maintaining connectivity.

In Section 3 we describe a *gossip-based* (e.g., [61]) protocol for *membership management* in dynamic overlay networks. List of members maintained at the nodes allow for information dissemination to all (or to a subset of) the peer nodes, and can be used in turn for applications involving distributed calculation of statistical measures of large set of numbers (see Section 5) system monitoring (see Section 6), as well as load balancing, which is discussed in BISON Deliverable D08. Our protocol is intended to build and maintain an overall overlay topology, that can be also called "knows-about", on the basis of partial network views maintained at each node. Unstructured knows-about overlay networks are analogous to *social relationship networks*, biological relationship networks like *food chains* or simply a physical proximity topology like the one defined by *cells* in an multicellular organism. Building this biological analogy further, the typical functions that can be implemented based on these topologies are *epidemic broadcasting* [56] and *diffusion*.

For what concerns MANETs, we have studied an algorithm based on an integer programming formulation for solving *minimum power connectivity problems* (Section 4). That is, we are addressing the problem of controlling the radio emission power at each node in order to minimize local power consumption while maintaining the conditions for global network connectivity (in terms of both the possibility of reaching every node in the network and minimizing in practice collisions when accessing the shared radio channel). While our initial model solves a centralized version of the problem, we are moving to an implementation which is both distributed and adaptive.

Collective computation: In Section 5 we discuss *aggregation*, which is the collective name of several functions that provide a variety of statistical information about a system [150]. Aggregation can provide members of a P2P network with important system-wide information. Furthermore, aggregation can be used as a building block for more complex protocols and functions (e.g., monitoring, described in Section 6). Our work on aggregation includes a robust and adaptive protocol for calculating aggregates in a proactive manner. The core of the protocol is a simple *epidemic-style* scheme based on the gossip-based protocol described in Section 3. We have derived several theoretical results concerning the convergence and the convergence rates of the protocol to the true global value, under the assumption of connectivity of the underlying overlay network.

Monitoring: We have used the techniques developed for collective computations to design a completely decentralized system for monitoring in overlay networks (Section 6). In

particular, we have applied the same epidemic-style protocols for the management of dynamic unstructured topologies (Section 3) for monitoring network size, measuring the total amount of resources and active agents, and distributing alarm signals in overlay networks.

Moreover, in Section 7 we present a study on the properties of *ant-inspired* routing systems to realize online *performance monitoring* in generic dynamic networks. Monitoring tasks in large networks have been usually carried out by means of passive data collection and offline elaboration. On the other side, routing algorithms based on mobile agents like those inspired by ant colony behaviors (see also Section 1 and Appendix A) can allow *active data collection* and *online monitoring*. We study the general properties of such systems (in particular of AntNet [37] and Cross-Entropy [79]) in order to identify which one of their components can be used as an adaptive indicator of what is going on in the network in the perspective of a use for distributed online monitoring tasks in generic dynamic networks.

Part I
Routing functions

1 Ant-inspired models for routing in MANETs

In this section we describe a set of algorithms for routing in MANETs. Their design has been inspired by shortest path and other organized behaviors observed in ant colonies. We discuss the characteristics of and the rationale behind this biological inspiration in the subsection that follows, while Subsection 1.2 discusses related work concerning both general approaches and ant-inspired algorithms for MANET routing. The hierarchy of the three ant-based routing models that we have developed so far is described and discussed in Subsection 1.3.

1.1 Biological inspiration

The general routing problem consists in the assignment of paths connecting sources and destinations of data flows maximizing some performance criteria. Therefore, in spite of specific differences that might arise because of differences in the characteristics of the transmission components, traffic patterns, performance metrics, etc., the routing component of a network always has to deal with the solution of *minimum cost path* problems. The constraints set of the related optimization problem to be solved is determined by the specific characteristics of the network at hand, where these characteristics include: transmission technology, delivered services, dynamics of topological modifications, statistical properties of the incoming data flows, etc. In the case of interest of MANETs possible solution strategies are required to be fully distributed and adaptive with respect to both topological modifications and traffic patterns. Moreover, since in general robust models for the distribution and the characteristics of the incoming traffic flows and mobility patterns are not available, an optimal routing approach cannot be pursued, and some suboptimal form of shortest path routing must be used.[1]

In the spirit of Bison objectives, we have searched for CASs in Nature which adaptively solve similar minimum cost path problems subject to constraints analogous to those imposed by MANETs characteristics.

CASs in Nature solving distributed minimum cost path problems

Our interest has been almost naturally attracted by ant colonies, which have already proved to be an effective source of inspiration for the design of effective routing algorithms [37, 134, 80, 46] by virtue of their ability to discover shortest paths between their nest and food reservoirs [66, 35]

[1] Optimal routing [13] has a network-wide perspective and its objective is to optimize a function of all individual link flows using a priori knowledge about their arrival distribution and data generation characteristics. Shortest path routing [13, 153] has an origin-destination perspective: the path between each node pair is considered more or less in isolation from the paths for all the other pairs. No a priori knowledge about the traffic process is required, although such knowledge can be fruitfully used, when available.

The shortest-path behavior shown by most species of ant colonies is the result of their foraging behavior based on pheromone trail laying/following [82]. While moving, individual ants deposit on the ground a volatile chemical substance called *pheromone*, forming in this way pheromone trails. Ants can smell pheromone and, when choosing their way, they tend to locally choose in probability the paths marked by stronger pheromone concentrations. Among the ants which have left their nest for foraging, the firsts arriving at the food source are clearly those which followed the shortest among the available paths. Accordingly, the pheromone that these same ants have deposited on these shortest paths while moving toward the food makes these paths, in the neighborhood of the food source, marked by more pheromone than the longest paths, not yet marked by any pheromone. The higher levels of pheromone present on these shortest paths stimulate these same ants to probabilistically choose them again on the way back to their nest. During their back journey, additional pheromone is deposited. In this way, pheromone is deposited on the shortest paths at a higher rate than on the longest ones, making the choice of the shortest paths more and more attractive for the subsequent ants at each local decision point. The experimental observation is that, after a transitory phase, most of the ants use the shortest paths with a probability which roughly increases with the difference in length between the different available paths and decreases with the number of possible alternative paths.

Pheromone acts as a sort of *dynamic collective memory* of the colony, a geographically distributed repository of all the most recent "foraging experiences" of the ants belonging to the same colony. By creating and sensing this chemical repository the ants can communicate and influence each other. Indeed, this basic form of indirect communication mediate by the environment, called *stigmergy* [68, 42, 146], can allow the colony as a *whole* to discover shortest paths in a completely *decentralized and distributed* way, as well as to give rise to other complex self-organized behaviors [22] (e.g., nest building in termites [67, 68]). [2]

General design principles from ant colonies

From a careful analysis of the collective behaviors observed in both ant colonies and artificial ant colonies (e.g., ACO), it is apparent that the generation of global patterns is the result of the interplay among a number of components, with pheromone and stigmergy being among the most important ones. In general, the *ant way* for the adaptive and distributed discovery of shortest paths features (e.g., see the results, both experimental and theoretical, reported in [22, 82, 35, 46]):

- pheromone variables, which express the goodness of each possible local choice and are the result of a continual collective learning process;
- stigmergic communication by means of pheromone variables;
- stochastic decisions;

[2] All these concepts have been reverse-engineered and put at work in the framework of *ant colony optimization* (ACO) [49, 48, 46, 44, 47, 50], which provides a formalization and systematization of these ant-based strategies and it is *de facto* the reference framework for ant-inspired optimization algorithms. The reader can find in the mentioned references in-depth discussions concerning the properties of these important class of ant-inspired algorithms which is also our main reference.

- incremental path generation;
- forward-backward path following;
- repeated and concurrent path sampling;
- population diversity;
- adaptive coupling between the characteristics of the individual ant and those of the local environment;
- fault-tolerance and robustness with respect to the actions of the single ant;
- proactive and reactive behaviors;
- point-to-point and point-to-many communication acts;
- diffusive pheromone fields with range-limited influence on the neighborhoods;
- natural decay of pheromone levels;
- globally adaptive response from locally adaptive behaviors;
- ant repertoires that allow the individual ant to build a complete path but such that near-optimal paths are expected to be the result of the collective learning dynamics.

These basic notions and strategies from the ant world have been already applied in some extent and with good success to solve routing problems (see following Subsection A.2).[3] For instance, this is the case of *AntNet* [37, 38], an ACO algorithm for adaptive routing in wired datagram networks which has shown better than state-of-the-art performance under extensive simulation studies. These facts confirm the basic intuition that these components at work for shortest paths discovery in ant colonies represent a really good match for routing problems.

We take these components as the basic building blocks to guide the design of our CASs for the routing function in MANETs. Accordingly, we say that we adopt an *ant-based* approach. Since we intend to develop state-of-the-art algorithms and not limit ourselves to proofs-of-concepts,[4] we are going to make use of only those ant-based components that, after being properly engineered, analyzed and tested, have shown the capability of improving algorithm performance. In this sense, in accordance with Bison's objectives, our target consists also in reaching a clear understanding of which ant-based notions/strategies can be really effective to design CASs for routing management.

[3] ACO algorithms, which feature most of these notions, have been applied also to many other optimization problems, both static and dynamic. The reader is referred to the reviews contained in [46, 44, 45, 42, 47, 50].

[4] It is in this spirit that we have chosen *Qualnet* [133] as a platform for developing and testing our algorithms. Qualnet is a top-end commercial simulation and development environment which provides realistic packet-level simulation, as well as reliable implementation of most of the state-of-the-art routing algorithms.

General properties of ant-based routing

The adoption of ant-based strategies, and, more in particular, of a design featuring *mobile agents* (the ants) and *pheromone variables* for repeated path-sampling and stigmergic learning of the sampling strategy, is likely to result in algorithms enjoying the following general properties:

- *local* adaptiveness synergistically resulting in *global* adaptiveness,
- *robustness* to agent failures,
- availability of both *multipaths* and *alternate paths* for data routing,
- automatic *load balancing*.

In particular, these behavioral patterns can be observed in *AntNet* [37, 38], which is a sort of prototype algorithm for ant-based routing, implementing most of (but not all) the basic strategies which are the fingerprints of the ant way. In order to develop our models according to the principles of ant-based routing, we precisely took AntNet as initial reference. AntNet was designed for wired datagram networks, therefore we have adapted the basic ideas behind AntNet to the MANET context, and we have further extended, improved, and enriched its design with the addition of several new strategies inspired by ant behaviors and/or matching the specific characteristics of MANETs. AntNet is discussed in Subsection A.2.

1.2 Related work

Over the past decade a vast amount of work has been done in the area of MANETs, and especially in the field of MANET routing. Clearly it is impossible to give a complete overview of existing work in this area. Instead, in Appendix A.1 we give a description of the most important algorithms, and give an overview of different research directions, so that the reader can get an idea of the context in which the research presented in this section is situated.

On the other hand, since our models are inspired by ant behaviours, we need also to provide an overview of existing work in the field of ant-inspired algorithms for adaptive routing in both wired and mobile ah hoc networks. As was pointed out in Subsection 1.1, ant-based strategies have properties like robustness, multipath routing, load balancing and global adaptiveness resulting from local behaviour. These features are important for routing, and especially for routing in MANETs, with their dynamic topology and data traffic, and their lack of central control or global overview. It should therefore not come as a surprise that already several ant-based MANET routing algorithms have been proposed. Most of these routing protocols are inspired either by Di Caro and Dorigo's *AntNet* [37], an ACO based routing algorithm for wired networks, or by the older paper [134] by Schoonderwoerd *et al.*, which proposes *Ant Based Control* (ABC), an ant-based load balancing method for circuit-switched networks, and was in itself an inspiration for AntNet. Appendix A.2 contains a description of some of the most important existing ant-based algorithms for routing in MANETs.

1.3 Models

Taking AntNet as initial reference and, more in general, using concepts from the ant-based approach as main guideline, we have developed a hierarchy of three routing models. Going from the bottom to the top of the hierarchy, we have incrementally added features which are more and more specialized for MANETs and enlarged the spectrum of the possible MANETs scenarios that could be effectively handled by the algorithm.

The design of the first of the three models in the hierarchy is directly based on the AntNet's architecture but introduces some important additional aspects specific for MANETs, like: *reactive* ants for session setups and management/advertisement of broken connections, proactive ants acting as *route maintenance* agents, *HELLO* messages to obtain fresh local information.[5]

The rationale behind the design choices for this first model, that hereafter we call *AHntNet*, was to have an algorithm: (i) based on the general ant-inspired principles, in order to get a clear understanding of these principles and of their possible role in a routing system, (ii) portable across networks with different characteristics, that is, usable without any major change in fixed, mobile, and ad-hoc networks, and, more in general, in *heterogeneous networks*. Moreover, since we needed to get a precise understanding about the implementation details of routing in MANETs, as well as to get familiar with the use of Qualnet, we have preferred in a sense to train ourselves starting with a model based on concepts which were quite general, well established, and familiar to us, in order to put the basis for and facilitate incremental successive developments.

This first model appears to be suited for small to medium-sized networks, and it is not expected to reach in general peak performance for specific sub-classes of scenarios, but rather to provide satisfactory performance over an extensive range of scenarios of both mobile and fixed, static and dynamic networks.

The second model, that we present in Subsection 1.3.2, is built on top of the ideas of the first one but adds several new concepts fitting some of the most peculiar characteristics of MANETs. In particular, this extended model, that hereafter we call *MAntNET*, introduces the important notion of *virtual circuits* through the explicit management of the path identifiers associated to the single traffic sessions. Therefore, while AHntNet provides *best-effort datagram* routing, MAntNET can be configured as a *reliable* datagram service which, by virtue of the use of virtual circuits, can offer an explicit control over the data flows, ensuring some level of quality-of-service, a more efficient management of multipath and alternate routing, and cycle-free routing.

Passing from the best-effort datagram model of AHntNet to the virtual-circuits one of MAntNET, amounts, from the point of view of the pheromone-laying model, to pass from a situation in which there is a unique and shared pheromone trail for all the traffic sessions between the same source-destination pair, to the situation in which every session has associated a different pheromone trail, specifically set up for the session. In the jargon of the ant metaphor, this appears to be equivalent to the situation in which *multiple colonies* are concurrently active and each colony makes use, as it actually happens in nature, of a pheromone trail having a

[5] It is interesting to notice that all these aspects added to AntNet can actually find their counterpart in the ant world. This is immediately understood for the case of reactive and proactive ants. On the other side, the HELLO messages, once intended as a way to spread information from one node into its neighborhood, can be configured as functionally equivalent to the process of local and range-limited pheromone diffusion which affects the pheromone field in the physical neighborhood of each point of the environment where pheromone has been deposited.

unique chemical composition. The whole pheromone on the environment is a shared resource for what concerns the path-discovery phase, but becomes a colony-specific resource when it comes to use the path for data, that is, when it comes to bring food from the food reservoir to the colony's nest.

MAntNET, with its MANET-specific features, has been designed with the explicit target of obtaining state-of-the-art performance for a wide range of small to medium-sized networks and traffic/mobility scenarios. However, it is not clear if it can be still efficient in *large networks* (from thousands to millions of nodes).

For these cases we believe that the use of *geographic information* is a sort of inescapable additional tool that the network has to be equipped with. Several already existing approaches make extensive use of GPS-based information (see Subsection A.1). Our third and last model, discussed in Subsection 1.3.3, goes exactly in this direction. The first two models, in order to establish a path for a new data session, make heavily use of broadcasting/flooding of reactive ants. This approach, which is also followed by the majority of state-of-the-art algorithms for medium-sized networks, does not appear suitable for large networks. In fact, flooding does not scale well. Unrestricted broadcasting in a large network would create a cascade effect that once repeated over and over could easily collapse the whole network. Therefore, in our last model, hereafter called *GeoMAntNET*, we use location information and the notion of *important nodes* (i.e., sort of gateways for either their relative position in the network or their knowledge in terms of paths to destinations), in order to look for a new path on directional basis. In this way we expect to narrow the search and dramatically reducing the amount of broadcasting while still obtaining satisfactory performance scaling up to very large networks.[6]

With our three models we expect to cover a really wide spectrum of different possible scenarios in terms of traffic patterns, mobility and topological characteristics of the network. Actually, our ultimate target consists in having at hand not several but one parametric algorithm able to adapt its internal parameters according to the detected/learned characteristics of the network, in order to locally switch to the general algorithm behavior which is the most appropriate for the characteristics of the network problem at hand. For instance, if the nodes realize that the network is large-sized then they will start heavily using geographic information, while for small networks it can be more appropriate and economic not to use geographic information at all. At this aim we expect to use Bison's results on monitoring and aggregation functions, in order to be able to make at the nodes estimates of network size and connectivity characteristics that could be used in turn to locally switch from one general behavior to another.

1.3.1 *AHntNet*: the basic model

AHntNet, the first routing model in our incremental hierarchy of models, is based on a strategy which mixes reactive and proactive approaches to provide best-effort multipath and alternate routing on the basis of datagram packet forwarding. The reactive part of algorithm is responsible for route setup when no route is locally available and for the management and advertising of locally broken connections, while the proactive components is intended for route mainte-

[6] This model too can be rooted in the ant world. In fact, for several species of ants, as well as other insects, when moving about a large foraging range, the most important sources of navigational information are celestial cues (e.g., sun position and sun-linked patterns of polarized sky light) and learned landmarks [156, 53, 54].

nance and additional route discovery. According to its ant-inspired background, the algorithm makes use of routing tables whose entries are pheromone variables, that is, variables expressing statistical estimates for the desirability of selecting a specific neighbor to forward packets toward a given destination. AHntNet does make use of only one pheromone value for each pair neighbor-destination, without making any further distinction among the routes associated to the different data sessions for the same destination, as the next model MAntNET does.

Both path-searching ants and data packets adopt a stochastic decision policy parametrized by the local pheromone values to decide which next hop to choose. This strategy provides the ants with an explicit exploration component and allows data packets for the same destination to be possibly spread over multiple paths. Moreover, even if the data decision policy is set in a way such actually only the best next hop is chosen, the other entries in the routing tables can be always used as alternate paths in case of broken connection for the current best next hop.

The global behavior of the algorithm is described in the following unnumbered subsections, each one discussing a different component of it. Figure 1 reports a high-level description of the algorithm behavior at each network node as a finite state machine diagram. The description of the actions associated to each state can be found in the following of the text.

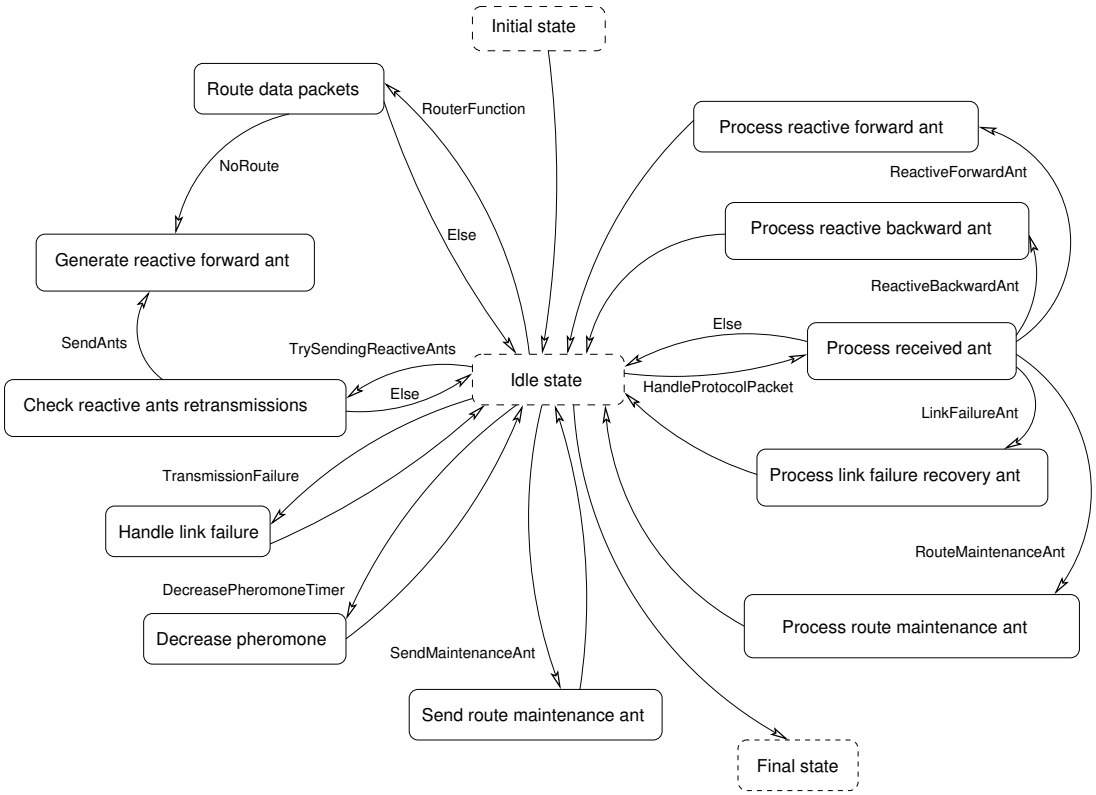


Figure 1: Finite state machine diagram representing the global behavior of AHntNet at each network node. The labels associated to the arcs represent the conditions or the events which trigger state transitions. Arcs without label mean that after the execution of the state actions the state transition associated to the arc is automatically executed. The "Else" transition is executed if none of the other possible conditions for state transition are met.

Information maintained at the nodes

At each node k the *routing table* $T(k)$ consists of a list of entries $T_{nd} \in \mathbb{R}$ that for each known neighbor $n \in \mathcal{N}_t(k)$ and destination d express the goodness of forwarding to neighbor n a data packet traveling toward the destination d. The T's values are also called *pheromone values* since they play the role of parameters of the stochastic routing policy and are the learning target of the algorithm.

As is explained in the following, after a path $k \rightarrow d$ has been found by a reactive ant, the routing table entries are updated in k to reflect this new knowledge. If the ant used the neighbor $n \in \mathcal{N}_t(k)$ to go toward d, the entry T_{nd} is either added to the routing table if such an entry was not already there, or it is updated. The pheromone value T_{nd} is updated (set) according to the *quality* of the followed path. In AHntNet, as well as in all our other models, the quality, and, therefore, the pheromone, is defined taking into account *multiple criteria* like: number of hops, traveling time, connectivity along the path, stability of the paths, etc. All these different but in a sense equally important components of path quality are put into a function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ to assign a new value to the estimated goodness of choosing n as next hop for destination d.

Evaluation metrics like the traveling time or number of hops cannot be evaluated on an absolute scale in a MANET. Due to mobility and non-stationary traffic patterns these variables are expected to show large fluctuations. Therefore, it is necessary to maintain at each node k tables $S(k)$ of statistical estimate for these measures. However, the ever changing conditions in the network would make any local statistics rather meaningless in a possibly very short time interval. Therefore, statistics are maintained only on *per data session basis*. That is, after the starting of a new data session $D_d(k)$ at node k, a list of statistics $S_d(k)$ associated to the data session is created and updated for the whole activity time of the session. When the session has completed its task the associated statistics can be cleared from the node memory.

Reactive ants for path searching

At the starting time of a new traffic session $D_d(s)$, the node s might have or not an entry in its routing table to forward the packets toward destination d. In the negative case a *route discovery process* is initiated. A *reactive forward ant* with d as final destination is generated. The task of the mobile agent consists in finding a path $s \rightarrow d$. Once the destination d is reached, the forward ant becomes a *backward ant* and retraces back its forward path, updating the routing tables along the visited nodes and creating in this way the conditions for the packets of the data session to be delivered.

The forward ant is usually *broadcasted* (see the discussion that follows), resulting in the generation of a broadcasting spanning tree, or network *flooding*.[7] On the contrary, backward ants are *unicasted*, since they do no need to carry out any additional exploration.

At the beginning of its journey the ant is a rather small packet, it carries: the IP number of the originating node s, a node-unique identifier seq for the traffic session for which it is looking for a path, and a vector of parameters which will characterize the ant behavior. The parameter values, withdrawn from a probability distribution common to all the nodes, specify the probability P_b of being further broadcasted and the probability P_k of being killed. Assigning the

[7] It is interesting to notice that each single broadcasting can be seen as equivalent to a sort of *proliferation* followed by uniform spatial *diffusion*. This behavior is commonly observed in biological *cells*.

ant characteristics by sampling from a distribution has a twofold effect: (i) we do not need to precisely define crisp thresholds, whose values might critically depend on the characteristics of the specific network at hand, (ii) some *diversity* is introduced within the ant population. These facts, are expected to favor global adaptability, robustness, and the generation of a richer repertoire of patterns and behaviors, as it happens in the case of a large number of CASs in Nature (e.g., see [57, 1]) and also in some of the artificial CASs studied so far (e.g., see the discussions in [91, 92, 7]).

Searching for a path by applying a flooding strategy is surely effective but it is not expected to be efficient because of the negative impact that might likely result from the large number of concurrent broadcastings executed by neighbor nodes. In the case of unrestricted and repeated broadcasting of the reactive ants at the nodes, *all* the possible paths $s \rightarrow d$ in the network would be eventually followed by the ants during the route discovery process. Therefore, some appropriate strategy has to be applied in order to reach a satisfactory balance between the ability of discovering good paths and the amount of routing overhead.

When a neighbor n gets the broadcasted ant, first it checks if it has already received an ant with the same (s, seq) values (this can easily happen because of repeated broadcastings). In the positive case, the ant is either discarded or further processed if the quality (in terms of number of hops and time) of the path followed so far scores favorably with respect to the paths of ants from the same route discovery process that have already passed by n. If after this check the ant has not been discarded, it is further processed. The ant is then killed with a probability which depends on P_k and local conditions (e.g., how many ants from the same process have been already forwarded, how many packets/ants are waiting in the node queue, etc.). If the ant survives this stage, then it will be forwarded according to a modality (broadcast, multicast or unicast) which is selected on the basis of both the value of P_b and other local conditions. That is, the ant internal characteristics and those determining the local state of the node are combined in order to obtain *locally adaptive behaviors*. For instance, if the node does not have a routing table entry for the destination, the ant could be broadcasted. However, if the ant has been already broadcasted several times during its journey, it might be better to avoid an additional broadcasting, and the ant could be either killed or unicasted. On the other side, if the ant has not been broadcasted yet but the number of neighbors of the node is large, it might be more appropriate to multicast the ant to a small subset of the neighbors, in order to reduce the amount of subsequent retransmissions from them.

In the case the routing table is being used to take the routing decision, all the existing entries for destination d are taken into consideration. A pseudo-random rule is adopted: with some large probability the entry for the neighbor h with the highest value, $T_{hd} > T_{nd},\ \forall n \in \mathcal{N}_t(k), T_{nd} \in T(k)$, is chosen, while with some small probability a random proportional selection is executed after normalization of the T_{nd} values.

When a reactive forward ants arrives at its destination, it reports the characteristics (traveling time and number of hops) of the followed path to the routing component of the destination node. Here, data received from all the forward ants belonging to the same route discovery process are stored. According to the data stored so far, the routing component acts as a *filter*: if the reported path characteristics are in some sense good with respect to what received so far, the forward ant is allowed to become a *backward ant*. Therefore, it will retrace its path back to the destination and update the routing tables at the visited nodes. Coming from node $n \in \mathcal{N}(k)$,

at node k the backward ant will add or update the pheromone value $T_{n\delta}$ for using neighbor n for all destinations $\delta,\ \delta \in \mathcal{P}_{k\to d}$, with $\mathcal{P}_{k\to d} = \{\delta_1, \delta_2, \ldots, d\}$ being the path followed during the forward phase from k to d.

Topological characteristics of established paths and multipath issues

As it has been pointed out, each forward ant arriving at a node k undergoes a strict quality-check. If the path it has followed so far does not score favorably with respect to the paths previously reported from the ants belonging to the same route discovery generation, the ant gets killed. This strategy is aimed at limiting the overhead due to transmission of routing packets and to reduce the risk of loop generation. As a result, most of the paths followed by the ants of a same route discovery process will be mutually disjoint or tree-shaped. Figure 2 shows the characteristics of the possible paths that can be obtained with such a strategy.

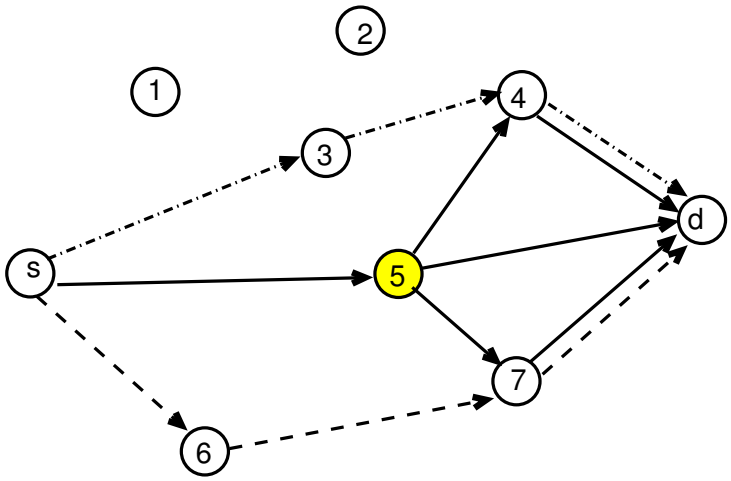


Figure 2: Topological characteristics of the paths that can be followed by reactive forward ants without being killed if the number of hops is the metric used for quality-checking. For instance, an ant arriving at 5 following either the path $s-1-3-5$ or $s-6-5$ would be killed since the ant which has followed $s-5$ has arrived first, and it has set the expected number of hops for quality-checking equal to 1. On the other hand, the broadcasted ant from 5 which has reached respectively 4 and 7 is not being killed since the paths $s-3-4$ and $s-5-4$ have same length, as well as $s-5-7$ and $s-6-7$.

Without filtering out the reactive forward ants during their journey toward the destination, and in the case the ants are broadcasted at each node (e.g., this can be the case when a path to a certain destination is searched for the first time), *every possible path* joining the source and the destination would be tried out by the ants. This would clearly generate an unbearable level of routing overhead, especially in the case of a large network. Moreover, the obtained paths would result intertwined, potentially giving rise to a number of loops. In fact, in the datagram routing scheme adopted in AHntNet, pheromone values from the path followed by the different ants paths might get combined such that loops are created. Figure 3 shows how this can happen because of the use of stochastic routing.[8]

[8] If the use of stochastic routing in a datagram network makes loops possible, it is also true that stochasticity

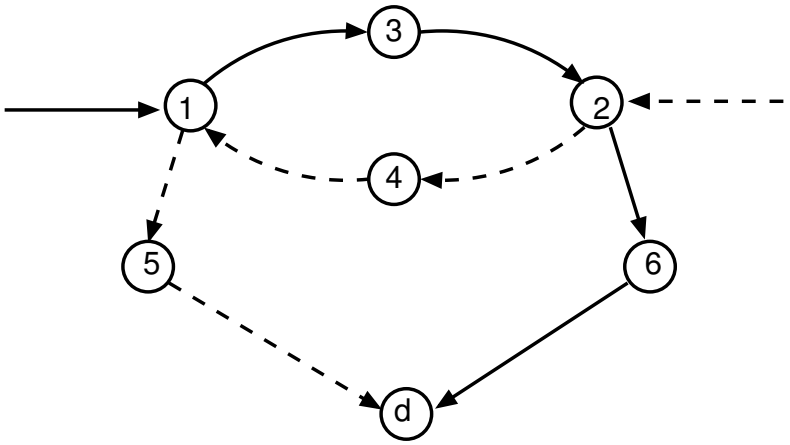


Figure 3: Combination of pheromone coming from different but intersecting paths followed by reactive forward ants can potentially generate loops. The risk of following a loop is in a sense intrinsic in a system using stochastic datagram routing. The figure reports the case of two forward ants which have followed two different but intersecting paths (showed by the dashed and solid lines) to reach the destination d. Let us assume that the amount of released pheromone is inversely proportional to the number of hops. Therefore, a data packet arriving in 1 will find a routing table with two entries for d: $T_{3d} = 1/4$, and $T_{5d} = 1/2$. If the data packet is routed according to a random proportional rule, it will go through 5 with a probability equal to $2/3$, while it will choose 3 with probability of $1/3$. Analogously, if it will move along the solid path, in 2 it will choose 4 with a probability of $1/3$ and 6 with $2/3$. Therefore, the data packet will loop between 1 and 2 with a probability equal to $1/9 = 0.11$.

There are also additional reasons to reduce the number of paths over which data for the same source-destination pair can be spread over. In fact, if the availability of *multiple paths* can be seen as an advantage in terms of load balancing, increased robustness, and possibly higher throughput, this might not be always the case in MANETs. To be effective, multiple paths must be: (i) *node disjoint* in order to provide higher robustness, (ii) *spatially disjoint*, in order to reduce the amount of radio interference (i.e., collisions) and provide *load balancing* ,[9] (iii) close, in terms of traveling time, to the path with the lowest end-to-end delay, in order to provide at the same time *higher throughput* without affecting the session end-to-end delay and creating potential problems in terms of packet reordering.

The filtering strategy adopted in AHntNet favors spatial and node disjointness, as well as the use of those paths with better end-to-end delay. However, it does not provide any real control over it. This is actually done in *MAntNET*, using the additional level of control provided by virtual circuits.

In spite of its limitations, AHntNet can actually provide at the same time *alternate* and *multiple paths* routing. This can be obtained by an adaptive tuning of the characteristics of the data

at the same time reduces the risk of being indefinitely trapped in a loop. Moreover, further updates of the routing tables by subsequent reactive or proactive agents are expected to eventually disrupt the formed loops.

[9] Multiple paths which are not far apart can be effective when data rates are rather low, such that packets from the same data session concurrently traveling over multiple paths do not really interfere over time.

routing policy: the parameters regulating stochastic data routing can be set such that only few best paths are concurrently used, while other paths are left as backup to be used only in case of necessity. For instance, this might be the case when the best paths become not anymore available because of node mobility. Again, this differentiation between paths that can be used to spread data sessions and paths that can be used as backups is made way more precise and effective in MAntNET by the use of virtual circuits.

Route maintenance ants

Once the first reactive backward ant comes back to the node s which initiated the route discovery process, the data session $D_d(s)$ can start sending packets. If after a defined time interval no backward ant comes back, the session is aborted. Once the session is active and sends data, *route maintenance ants* are generated from $s \rightarrow d$ in order to continually check the status of the paths that are going to be used by data packets. Every m data packets, where m is a parameter (e.g., $m = 10$ seems to be a reasonable choice), a forward route maintenance ant is generated at s directed to d.

The route maintenance ants behave like reactive ants with the only difference that their internal parameters are set such that they explicitly check the status of the best paths, that is, of the paths that are actually followed by data packets. Therefore, route maintenance ants are most of the time unicasted to the nodes which correspond to the best next hop entries in the routing tables. However, with some small probability P_e they are also broadcasted, in order to discover new and possibly better paths for the session. In this way, the routing overhead is in practice defined by the value of m as a percentage of the data flow.

The main purpose of route maintenance ants consists in: (i) *reinforcing* and *tuning* the pheromone levels on the best paths, (ii) *detecting* local route failures before data packets arrive at a node with a broken connection, (iii) carrying out some marginal *exploratory* activity in order to discover better paths and/or make new alternate paths available. The exploratory component of route maintenance ants is marginal since exploration (in terms of broadcasting) can potentially cause a large overhead, therefore, it must be kept as much as low as possible. On the other hand, path exploration is carried at the setup time of each data session by means of route discovery processes. The duty of route maintenance ants is to guarantee the same existence of connection paths for the data session while adapting routing tables without causing unnecessary overhead. Some form of proactive exploration is carried out also by means of *hello messages*, which are discussed later.

If a route maintenance ant cannot be unicasted to the selected next hop because of transmission failure, the next best available neighbor in the routing table is tried out. The process is iterated until no there are not anymore entries in the routing table. If the ant cannot be successfully unicasted, the ant is killed. In this case, as well as, in the case the transmission failure concerned the best next hop in the routing table, a *route failure recovery process* is initiated.

Recovery from broken connections

When either a data packet or a unicasted ant cannot be sent to the selected next hop node f (e.g., the node has moved away, has been turned off or emission/transmission power has lowered), a *route failure recovery process* is started in order to update the routing tables to the

new topological situation. In order to recover from the route failure situation, AHntNet adopts a mixed strategy: *local advertising* of the route failure and generation of reactive ants for *local repair*.

After getting a transmission failure, the next hop node f is considered as not anymore belonging to the current neighborhood $\mathcal{N}(k)$ and all the related entries in the local routing table are removed. If, after this operation, some known destinations have not anymore a next hop entry in the routing table a route discovery process is initiated for each one of these destinations in the attempt to repair the routes.

In the case all known destinations still have a next hop entry in the routing table, there is no need for new route discovery ants, but it might happen that a set D' of some destinations have lost in this way their best next hop connection while the remaining connections score poorly in terms of pheromone levels with respect to the lost connection. In this case, it might be helpful to tell the other nodes which were relying on the pair (k, f) to reach the destinations in D' that the connection between k and f is broken and that the remaining connections are not equally good. At this purpose, *route failure ants* are created and broadcasted. The broadcasted packet contains for each destination d' in D' the new best number of hops $h_{d'}(k)$ to reach it from k as it results from the values stored in $S_{d'}(k)$. The ant failure packet is received by k's neighbors and for each $d' \in D'$ is treated as it were a sort of ant which had traveled up to d' in $h_{d'}(n) = h_{d'}(k) + 1$ hops using k as next hop, with $n \in \mathcal{N}(k)$. The routing tables of each node n which have received the ant failure are updated according to the new values $h_{d'}(n)$ which replace the old ones stored in $S_{d'}(n)$. If some destinations do not anymore have a next hop connection in k a symbolic value indicating an infinite number of hops is reported. After updating its own routing table, each neighbor in turn broadcasts an ant recovery failure if the conditions of either lost connection or lost of all best connections are met. In this way, every node propagates its updated view in terms of reachability of destinations and estimated number of hops. The process is expected to rapidly stop at few hops away from the originating node k. In order to reduce the overhead due to the repeated broadcasting of the ant failure packets the threshold to decide when it is worth to advertise the lost of the best connection must be put reasonably high.

In the case of MAntNET, it will be possible to directly advertise also the source nodes of the sessions which were using the broken connection for their virtual circuits. In this way the local repair can be effectively combined with a sort of global repair.

HELLO messages

The use of *HELLO messages* is quite common in MANET routing since it helps to get updated views of the node set in the current neighborhood. We use them in an extended form: nodes periodically broadcast a HELLO packet containing the node identifier and the list of the identifiers of its known neighbors and destinations.

The extended hello messages are used in AHntNet to *intercept route failures* and to obtain information about the *local network topology*. Every time a hello message is received, the neighbor node is either added as a new entry in the local list of neighbors or the timestamp of its list entry is updated with the current time, as well as the list of its neighbors and destinations.

In this way, before using a next hop node n to transmit a data packet, its timestamp is checked,

and in the case it is too old with respect to the periodic broadcasting interval, n is removed from both the neighbor and routing tables and an alternative action is executed.

The knowledge on the identifiers of the current neighbors and their known destinations is exploited to carry out *proactive exploratory actions* without repeated broadcasting. In fact, both reactive forward and route maintenance ants can be either unicasted or multicasted toward those neighbors which claim to have either connections for the searched destination or the best level of connectivity.

Moreover, the knowledge of the number of neighbors can be used to make raw estimates about the local size and connectivity of the network. As it has already pointed out, this information, can be used to automatically switch from geographic to non geographic routing and vice versa, that is, to adapt the routing strategy to the network topological characteristics.

Preliminary results

At this stage we have mainly focused on reasoning on the different possible ant-based strategies that could be fruitfully used. We have made some initial experiments using different realizations of AHntNet and we have compared its performance with state-of-the-art algorithms like AODV [122]. Performance appear as promising, at least in the highly connected and medium-sized scenarios which are usually considered in literature. Since we intend to use AHntNet in mixed-type networks, we have also checked its performance in scenarios without mobility, but with point-like topological modifications and for large wired IP networks.

1.3.2 *MAntNET*: the extended model

Network routing protocols can generally work in two different ways: connectionless or connection oriented. Connectionless routing is called datagram routing, and connection oriented routing is called virtual circuit routing. In datagram routing, each node has a routing table indicating how to get to the destination, and data packets are forwarded from node to node. This means that for each packet a new routing decision is taken in every node. Each packet is routed separately, and can in principle follow a different route to the destination. Datagram routing is normally proactive, and a good example of datagram routing in MANETs is DSDV. Virtual circuit routing is normally reactive: a connection is set up between source and destination at the start of a communication session, and every data packet follows this connection. This is the mechanism used in AODV: at the start of the communication between two nodes, a virtual path of pointers is set up between the source and the destination, and all data packets follow the exact same path.

The ant-based routing algorithm like AntNet uses datagrams. Each node proactively keeps a stochastic routing table to each destination, and routing decisions are taken independently at each node. For MANETs, however, we know from comparative tests in literature [20, 30] that proactive routing does not scale well. Nevertheless, datagram routing has some advantages which are quite important in MANETs. One of these is that datagram routing allows different data packets to follow different paths, which means that traffic load can be spread over the MANET, possibly leading to higher throughput and lower congestion. Another advantage is that thanks to the multipath nature of datagram routing it is possible to adapt quickly in

case of link failure. In order to keep these advantages, we have proposed in AHntNet a reactive version of datagram routing (plus some additional proactive mechanisms): routes are still indicated by different stochastic routing tables in each node, and routing decisions are taken from node to node, but the routing tables are only set up when a communication session between source and destination is started. A similar scheme can also be found in other MANET adaptations of AntNet, like ARA [71, 72] and PERA [10].

The scheme proposed in AHntNet also conserves an important disadvantage of datagram routing though, namely the fact that it is less reliable: since routing decisions are taken independently at each node, there is less control, and things can easily go wrong. For MANETs, where the network situation is very dynamic, it is in particular important that all nodes have correct up-to-date routing information. If one node has new information, and another node has old information, the combination can very easily cause loop formation (this phenomenon was actually observed in the proactive multipath MANET routing algorithm TORA [120, 20]). Having a reliable routing algorithm is in MANETs maybe even more important than in traditional wired networks, due to the fact that there are unreliable protocols at the lower layers (wireless communication at the physical layer, and a fairly unreliable MAC protocol at the datalink layer) as well as at the higher layer (UDP is used at the transport layer[10]). One way to make datagram routing more reliable is to increase the frequency of updates of the routing tables, but this would result in a badly scaling protocol. Another way could be to only use one route available in the datagrams, and not mix old and new information (which is done in the ant based routing algorithms ARA and PERA mentioned before), but then the algorithm is not very different anymore from virtual circuit routing, and looses the advantages of datagram routing.

Virtual circuits

In MAntNET, we try to build a more reliable routing algorithm while still keeping the advantages of data traffic spreading and quick adaption to link failures. In particular, we propose to use multipath virtual circuit routing with stochastic spreading of data traffic over the multiple paths. Like in AHntNet, paths are set up in a reactive way at the start of a communication session, but instead of building stochastic routing tables at each node, we set up a number of full paths between source and destination. The data packets will therefore only follow fixed paths which were completely tested by a single ant (this is not the case in AHntNet, since there a followed path is the result of a series of separate routing decisions in different nodes, based on pheromone which could be deposited by many different ants) and which are completely known at the source. This should make the routing much more reliable. Nevertheless, at the source there will still be a choice among several paths, and the source node will route data packets stochastically over these paths. This means that data traffic spreading and quick reaction to failure is still possible. The main difference with AHntNet is that the stochastic decision is taken once at the source, and that afterwards the data packet follows a fixed path.

Apart from improving reliability, we also expect some other advantages from this new approach. Most importantly, the algorithm gives the source nodes better control over the paths which are used. Since each path is fully known at the source, it will be possible for the source node to select just a few good paths for data traffic. In particular, we want source nodes to select a number of fully disjoint paths. This can improve both traffic load spreading (if non-

[10]The more reliable TCP algorithm can due to several reasons not be used in MANETs (see [6, 63, 141]).

disjoint paths are used, data traffic on different paths can interfere with each other, cancelling out the advantage of the multipath routing, as it as already discussed at Page 19) and failure robustness (because several non-disjoint paths can fail at the same time if one of the nodes they have in common fails). Also, for each of the selected disjoint paths, we want the source to keep some backup paths. These don't need to be completely disjoint; they are meant to replace the primary paths in case of failure. Other advantages we can see for the new approach is the fact that in a later stage it will allow an easier implementation of advanced techniques like *quality of service* routing and *label switching* [2].

State information messages

In AHntNet we have described the use of hello messages to increase the nodes' awareness of the local network situation. Each node periodically broadcasts a small message, containing just its own address. Keeping track of received hello messages, nodes can derive information about network topology and mobility. A node can realise how many and which neighbours it has, and how stable the links with these neighbours are. It can also detect link failures and adapt its routing table according to them.

In MAntNET, we plan to keep the hello messages, but propose to also use a second kind of messages, called state information messages. The aim of the state information messages is to make route exploration less blind. In a state information message, a node broadcasts the paths it is on, and the length of these paths (the number of hops between the current node and the destination). Also, nodes include which paths its neighbours are on, and the lengths of those paths. All this information together should give each node a clear view of all the paths in its environment, and help it spot new routing opportunities. In particular, the state information messages should help nodes find alternative routes for existing paths (used for backup), find shorter versions of existing paths (for path improvement), and restore paths after a link failure. This is shown in parts a), b) and c) of Figure 4: a node which knows from the state information messages that its neighbour has a neighbour which is on a good path to the destination, can make an alternative or shorter version of the path it is carrying. In the spirit of virtual circuit routing, however, we do not allow a node to make the path alteration or improvement without properly setting up the new path between source and destination using a forward and a backward ant. Other information found through the state information messages is the interference between paths, as shown by part d) of Figure 4: if a node receives information about a different path going to the same destination over one of its neighbours, it can send this information back to the source. The source then knows that the two paths, even though they are completely node and link disjoint, can still interfere with each other, and should therefore maybe not be used together for data traffic spreading.

It is clear that state information messages will be quite large, and may cause a lot of extra overhead. One way to reduce this overhead is to send the messages less often than hello messages. In particular, we want to adapt their sending rate to the traffic (data and ants) on the paths, so that information about rarely used paths is not broadcasted very often. For the detection of interference between paths, this scheme seems very natural: rarely used paths give less updates about their presence, and interference with them will therefore be more tolerated than interference with heavily used paths. Another way to reduce overhead caused by state information messages is to add them on top of hello messages, so that only one big packet is sent,

rather than two small ones. Finally, in future extensions of the algorithm, described in Subsection 1.3.3, we hope to be able to completely stop the use of state information messages: it should be possible to use location information of the nodes to perform the functions for which we now need the state information messages.

Reactive ants

For path setup we use reactive ants, more or less in the same way as in the basic algorithm. When a communication session is started, a forward ant is broadcasted from the source, and flooded over the network. Intermediate nodes limit the flooding by deleting ants which are following paths that are clearly worse than paths followed by previously received ants. While going to the destination, forward ants keep a list of the nodes they have visited, and when they reach the destination this list is copied into a backward ant. At this point there are two differences with the basic algorithm. First of all, the destination node will make a selection among the ants to send back. The algorithm is not really interested in paths which are much longer than already established paths, or in paths which have a large degree of overlap with other paths, so the destination can delete these ants. This should reduce a bit of overhead. The second difference is that the destination will assign a path ID number to the backward ant, which will be used to distinguish its path from other paths. Just like in the basic algorithm, the backward ant is then unicasted back to the source over the same path over which the forward ant came. The backward ant will at each intermediate node set up the pointers for the path ID it is carrying.

Path selection

After the reactive path setup phase, the source will be presented with a set of possible paths towards the destination. It will select a few of these to use for routing. The selection criteria are disjointness, trip time, and hop count. All of these metrics are reported back by the backward ants which set up the paths. Disjointness is the most important criterium. As was pointed out earlier, disjoint paths give a more robust connection, since they are not dependent on the same

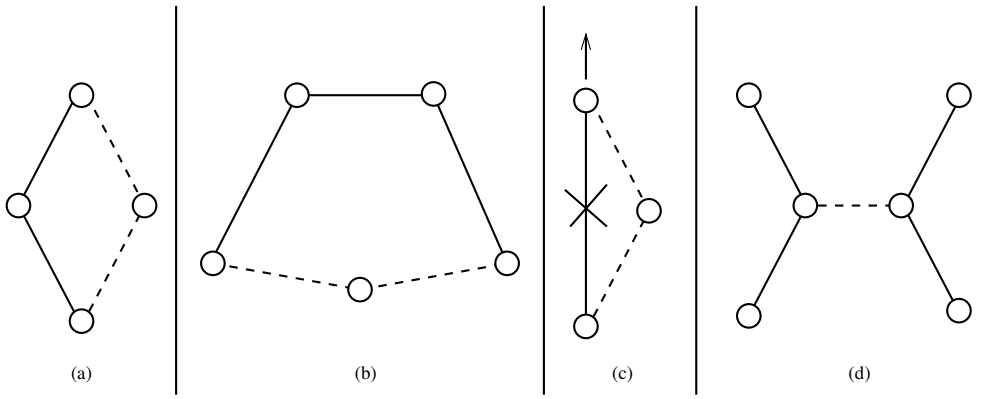


Figure 4: Using state information messages to (a) discover backup routes (b) find route improvements, (c) handle link failures and (d) detect path interference.

nodes: if one nodes moves away or switches off, it cannot disconnect all paths at the same time. Also, if paths are disjoint in the sense that there is not even radio interference between them, they will allow better data traffic spreading: traffic over one path will not affect traffic over the other path. Apart from disjointness, we also look at trip time and hop count. Trip time is an indication of the path quality: we want to send our data to the destination with minimum delay. However, in MANETs trip time can be very unstable (it can be changed very much by the presence of interfering traffic). Therefore we want to combine the trip time measure with the hop count, which is an approximation of the path quality: a path which is short in number of hops will normally also not be too long in trip time (or at least has the potential of being short in trip time, if the interference it is subjected to is low).

Path maintenance

As was pointed out earlier, the use of state information messages allows nodes to detect variations and improvements to paths they are carrying. The nodes can then send an ant to test the new path, and notify the source about it. The source can decide whether or not to accept the new path. In this way, each path of the original set of paths can be supplemented with a number of variations and improvements, so that it becomes a bundle of largely overlapping paths. The complete mechanism of using a set of disjoint paths with a bundle of variations for each of them should provide a very robust connection: minor disruptions can be dealt with easily by switching to one of the variations within the bundle, while major disruptions should be handled by (possibly temporarily) switching to one of the disjoint path bundles. Some support for these ideas can be found in [117], which presents a multipath version of DSR. The paper indicates that the algorithm's performance improves considerably when multiple paths are provided. The paper also points out that it is important to not only provide extra paths between source and destination, but also between intermediate nodes and the destination. The authors also find that just a few backup paths per node are sufficient to obtain the improvement.

Path monitoring and data routing

For the actual data routing, the source will choose from among the disjoint paths. The own variations of each path (as described above) are not used for data traffic, only for backup purposes. This is because it does not make sense to use largely overlapping paths for multipath routing: different data packets would eventually end up in the same area, competing for the channel with each other, and the situation would be worse than had they just been sent over the same path.

The data traffic will be spread over the different paths stochastically, using pheromone values which indicate the goodness of the paths. To obtain the pheromone values, the source node regularly sends out ants to sample the different paths. The ants report back the trip time over the path they followed, and this trip time is used to update the pheromone. The idea is that the trip time is defined by both the length of the path and the congestion in the areas which are crossed by the path. By probing the paths with the ants the source node can get an idea of this congestion, and can adapt its own data flow to avoid congested areas. This adaptation of the data flow will then change the congestion levels experienced by the other nodes. They

will in turn realise the change using their probing ants, and adapt their pheromone and their data traffic to it. The hope is that the collective adaptive behaviour of the different nodes will give rise to a globally (near-)optimal spreading of the data load over the MANET. One possible danger is that we would adapt the traffic spread too quickly, giving rise to oscillations in the network. The fact that trip times can be quite unstable in MANETs adds to this possible problem. It will be important to update the data traffic spread gradually.

1.3.3 *GeoMAntNET*: further extensions for large networks

Routing in wide area MANETs (WAN MANETs) is still a relatively unexplored research area. Most MANET routing algorithms are developed for small networks, and even though some papers claim to propose a well scaling algorithm, they do not usually present simulations of more than 50 or 100 nodes. An exception is a scalability study of AODV, which goes up to 10000 nodes [101]. It shows a very quickly degrading throughput though (it is down to 50% at 1000 nodes), due to the long path lengths. Some of the few algorithms specifically developed for WAN MANETs are the earlier mentioned Terminodes project [14, 16, 15] and MABR [78].

In what follows we present some extensions to our algorithm to make it more suitable for WAN MANETs. In particular, we present adapted schemes for path discovery, using *anchored paths*, finding critical links, selecting disjoint paths, and improving known paths. Throughout this subsection we will assume that nodes know their own location coordinates (e.g. through the use of a GPS system). This extra information is quite essential for routing in WAN MANETs. It allows for simpler control messages, and more robust routing.

Path discovery

In the routing algorithms described so far, we use flooding for path discovery: the source node broadcasts a forward ant, and it is further broadcasted by the neighbors, until it reaches the destination. It is clear that such a scheme produces a lot of overhead. This can be more of a problem in WAN MANETs, since it was found in [74] that the transport capacity per node in a MANET is inversely related to the total number of nodes. Therefore we want to propose a more clever scheme for route acquisition.

One way to make the path discovery easier is to provide some *structure* in the network. This is discussed at Page 92 in Appendix A.1, where it is argued that it is better to have a neighbour selection protocol, were a structure emerges from the individual views of the nodes, rather than a partitioning protocol, where an overall structure (like a cluster structure) is imposed on the network. An example of a neighbour selection protocol is the friends mechanism in Terminodes [15], where each node actively searches a number of friends, which it thinks will be helpful in finding destinations. To these friends the node maintains good paths, so that it can always find them easily. Then, when a node needs to find a certain destination, it forwards its route request to the friends, which forward it in turn to their friends, until the destination is found. The friends graph is structured in such a way that it resembles a small-world graph, so that the search in the overlay friends graph can find any destination in a limited number of hops.

We propose to use a quite similar mechanism, with an *overlay graph* used for path acquisition. Rather than letting each node choose its own friends though, we would like the friends to be found automatically, as a result of the network topology. Some inspiration for this can be found in subgoal discovery in reinforcement learning. In subgoal learning (see [143]), the idea is to help the learning of goal-finding tasks by learning subtasks which could be useful in many different situations, just like for humans the subtask of walking to a door can be useful in tasks ranging from going to the railway station to going to bed. A subgoal discovery algorithm has to automatically find certain areas in the agent's world where it is useful to temporarily leave the search for the goal and instead search for the subgoal. Once the subgoal is reached, the agent can go back to trying to find the overall goal. The process of discovering good subgoals and learning how to travel to them is done beforehand. At the starting time, random start and goal states are given, and the subgoals are used in the learning process. The subgoals should be useful at helping to solve any possible task in the agent's world (any combination of start and goal state). In [112], it is stated that bottleneck states are useful subgoals, because if an agent can get to these states more easily (since it has learned beforehand how to do it), it will much more easily explore the whole area of the world behind it. In [138] bottlenecks are found by examining which states are visited most often. In this paper it is also found that learning about often visited states is useful even in environments where there are no bottlenecks.

It might be possible to use something similar in WAN MANET routing. Nodes acting as major gateways (logically equivalent to the above bottlenecks) will appear in the routes of all ants which travel from one side of these nodes to the other. Also other *important nodes*, like very stable nodes, nodes with a larger range, or very central nodes will be present in a lot of good routes. Therefore, nodes should look out for nodes which are more often present in routes than other nodes. Once these important nodes are recognized, they can be used like the friend nodes: when nodes do not have any idea about where to find their destination, they can send their forward ants to the few important nodes they know about. Ants arriving in the important nodes should have a double advantage. First, since they have traveled to a node which is on many good paths, they have a larger chance of being on a good path to their destination. Second, they arrive in a region where many paths converge, which means that there is a lot of information available there. So they have a larger chance of finding information about their destination node. Of course, while going to one of the important nodes, ants should also look out for their destination. Also, it should be possible to let ants fork at intermediate nodes, when at these intermediate nodes new important nodes are known. A possible danger in this approach is that some nodes get much more traffic than other nodes, and that, especially if these specific nodes are already bottleneck nodes in the topology, fragile areas of heavy congestion will be created.

Anchored paths

The routes returned by the forward ants are lists of visited nodes, with their location. In WAN MANETs these lists can be very long. We would like to adopt the idea of anchored paths used in Terminodes [15], where paths are indicated by a short list of key locations. Packets following the path roughly go from one key location to the next, using greedy geographic forwarding between them. Apart from making the path information shorter, this scheme also makes the paths more robust: paths which consist of a list of node IDs break as soon as one of

the nodes moves too much, whereas geographic forwarding between anchor points can easily use different intermediate nodes (if at least there are not too many gaps in the network). This property makes the use of path variations (discussed in 1.3.2) unnecessary, and can reduce the need for state information messages.

Critical links

WAN MANETs are likely to have a *varying node density* over the whole network. In [52, 51], it was indicated that in low density MANETs connectivity can be very low, depending on just a few *critical links*. If these critical links break (due to node movement e.g.), backup paths will often have to be found over completely different parts of the network. This can be a slow and difficult task. In the extended model presented above, we account for this by supporting setting up and maintaining multiple disjoint paths, hoping that their disjointness will imply that they do not use the same critical links. In GeoMAntNET we want to go a step further. It should be possible for a node to realize that it is on a critical link. Using location information in the hello messages, it could figure out whether one of its neighbours is not in contact with any of its other neighbours. Or a node could figure out whether there is one neighbour which is carrying an unusually high percentage of the traffic, or whether the paths going over one neighbour visit parts of the network which are not visited by paths going over the other neighbours. Once a node realises it is on a critical link, it can send this information, together with its location, to the source nodes from which it receives packets. These source nodes can then explicitly look for paths which do not go near the same critical link.

Selecting disjoint paths

In MAntNET, we use multiple disjoint paths between source and destination. There were two reasons why the paths had to be disjoint: to increase robustness, since disjoint paths usually have uncorrelated failures, and to allow for a better spread of the data load, since there is no interference between the different paths (we choose the paths so that they are not only node-disjoint, but also radio-disjoint). Although making different paths avoid using the same critical links (as explained in the previous paragraph) should go a long way in providing the increased robustness, it does not assure that there will be no interference between paths. In MAntNET we use state information messages to detect which paths can interfere with each other. However, when using location information, it should be possible to avoid the use of these messages. Since the source node knows the location of the different anchor points of the path, it knows more or less where the packets will pass, and can judge whether two paths will be sufficiently far apart to be radio-disjoint. So it can select disjoint paths without needing the state information messages.

Finding path improvements

A last task we use the state information messages for is to find path improvements. In GeoMAntNET location information could replace the use of the state information messages. Since for each path a number of successive location points are known, it is easy to think of path improvements: any path which uses a subset of the current path's anchor points is a possible new path. Also combinations of anchor points of different paths can be used, or two successive

anchor points can be replaced by one point in between them. It is easy for a source node to calculate whether the possible alternative paths are at least in theory shorter than the current path. It can then send ants out to test candidate path improvements.

Part II
Search functions

2 Search in P2P networks inspired by concepts from the immune system

In this section, we report an algorithm for searching p2p networks. The fundamental ideas of the algorithm has been stolen from natural immune systems known for using robust, decentralized algorithms to kill diseases. The algorithm avoids query message flooding, instead of which employs concepts of proliferation and mutation. These concepts are directly borrowed from the immune system. Moreover, the topology of the network is changed during the search resulting in clustering of nodes hosting similar items. The entire algorithm is presently implemented on grid topology although it can be implemented in any other type of topology. The basic model parameters and the algorithm is next presented.

2.1 Biological inspiration

The natural immune system is a complex adaptive system which efficiently employs several mechanisms for defence against foreign pathogens. The main role of the immune system is to recognize foreign pathogens and to induce an appropriate type of defence. From an information-processing perspective, the immune system is a highly parallel intelligent system. It uses learning, memory, and associative retrieval to solve recognition and classification tasks. Its general features provide an excellent model of adaptive processes operating at a local level and of useful behavior emerging at the global level.

Basic ideas of Immune System used in our problem: We have used the simple and well known concept of humoral immune system where B cells generate antibodies which tracks the antigens or the foreign body. In our problem, the query message packet is conceived as antibody which is generated by the node initiating a search whereas antigens are the searched item hosted by other constituent members of the p2p networks. Like immune system, the search undergo proliferation, activation, induction, differentiation, attack and memory formations.

The key features of the natural immune system based upon which the *ImmuneSearch* algorithm develops is noted next:

- *Recognition:* The immune system can recognize and classify different patterns of interest and generate selective responses. [In our problem, the message query packets which are generated for search emanate a varied type of response while visiting other peers in the network.]
- *Learning:* The immune system learns by experience. The learning ability of the immune system lies primarily in the recruitment mechanism [12] which generates new immune cells on the basis of the current state of the system (also called *clonal expansion*). [Our system learns through experience, so as time passes, the search efficiency increases.]

- *Distributed Detection and Self-Regulation:* The immune system is inherently distributed and the mechanism for immune responses are self-regulatory in nature. There is no central organ that controls the function of the immune system. [The search mechanism in our system is not guided by any centralized control. On the contrary, the query message packets within the network regulate their movement in a self organizing fashion.]
- *Diversity:* It uses combinatorics (partly by a genetic process) for generating a diverse set of lymphocyte receptors to ensure that at least some immune cells (lymphocytes) can bind to any given (known or unknown) antigen. [The query message packets undergo mutation to look for interesting search items in addition to the item that has been queried]
- *Threshold mechanism:* Immune response and the proliferation of the immune cells takes place above a certain matching threshold (Strength of chemical binding). [The query message packet undergo proliferation and mutation in the network only if it meets some search items which matches with it above some threshold.]
- *Aging:* As organisms grow old, their acquired immunity reaches a stable state and changes only if there is a radical change in the environment. [Each peer in the network stabilizes its position within a period of its entrance in the system. It only changes position if there is instability in the system.]

2.2 Related work

In this section, we note the related works which has been inspired by immune systems and the works which has been done to perform efficient search in p2p network.

Works inspired by Immune System

There are a growing number of intelligent methodologies (inspired by immune system) solving real-world problems. The methodologies map their problem to immune systems and accordingly use various features of natural immune systems to develop their respective algorithms. These methods labelled as Artificial Immune Systems, Immunity-Based Systems, Immunological Computation etc. have been used to model diverse applications namely searching mines [135, 136], anamoly detection in time series data [33], pattern recognition [84], virus detection [94] etc. A good overview of the various applications developed using immune system concepts is available in [32, 96]

Works related to search

The search algorithms should maximize the goals of efficiency in terms of resources (bandwidth, storage facility, processing power), quality of service (number of results, response time) and robustness. Side by side, it should provide the peers more autonomy (in terms of with whom it wants to be neighbor and/or in which machines it is ready to share its content) and expressiveness (how differently can a user express his search). Different protocols which have been developed are trying to maximize these two conflicting interests. A brief sketch is provided.

There are highly structured P2P networks, such as CAN [125], Chord [137] , Past [129] , and Tapestry [81], all of which use precise placement algorithms and specific routing protocols to make searching efficient. Most of them use Distributed Hash Table which has become quite

popular [70]. The hash-table-like lookup() operation provided by DHTs typically requires only O(log n) steps, where n is the number of nodes. However, although it is quite fast but its ability to operate with extremely unreliable nodes has not yet been demonstrated. Moreover, the systems built upon distributed hash table cannot deal with partial-match queries (e.g., searching for all objects whose titles contain two specific words).

In comparison, the loosely structured P2P networks, Freenet [59], Free Haven [40], Mojo Nation [111], are extremely robust but the search time is quite slow. Flooding and random walk are two of the methods used to perform search in these networks. In order to improve the performance, in unstructured networks there is a plethora of works related to some replication mechanism of data, so that the data can be found fast. There is Freenet which replicates data as soon as they are searched. The replication mechanism also can follow different strategies like proportional replication, uniform replication and square root replications [106]. However, none of the replication mechanism is still not provably considered as the best. Moreover, replication mechanism infringes on the autonomy of the peers, forcing them to host materials against their wishes.

Besides this, there are works on semantic overlay networks (SON) which proposes clustering nodes on basis of the same semantics [132]. That is, nodes having identical semantics forms neighbours to each other. The semantics can consider the network to be a flat structure [27] [145] as well as a super-peer structures [105]. There are many challenges when building SONs. First, we need to be able to classify queries and nodes. We need to decide the level of granularity for the classification (e.g., just rock songs versus soft, pop, and metal rock) as too little granularity will not generate enough locality, while too much would increase maintenance costs. We need to decide when a node should join a SON (if a node has just a couple of documents on rock, do we need to place it in the same SON as a node that has hundreds of rock documents?). Finally, we need to choose which SONs to use when answering a query. Of course, there can be queries which doesn't conform to a particular query, but can imply different SON at a time.

Keeping the above developments in mind, we propose to build a system which will provide autonomy to the user in terms of his storage right. It will also not as strict as SON. It will be in the main unstructured. However, it will take the clustering concepts of SON, but not try to form tight clusters like SON. Rather, it will allow the system to evolve on its own and the environment to decide which are the factors based upon which the self organization/clustering should be undertaken. This is in spirit of the biological adaptation policy where biological/natural processes self organize them according to the need of the surroundings.

2.3 The model of the P2P environment

The parameters which are important for simulating p2p environment are the network topology (overlay network), the profile management of each individual nodes, the nature of distribution of these profiles and the affinity measure based upon which search algorithm is developed. Each of the parameters are discussed one by one.

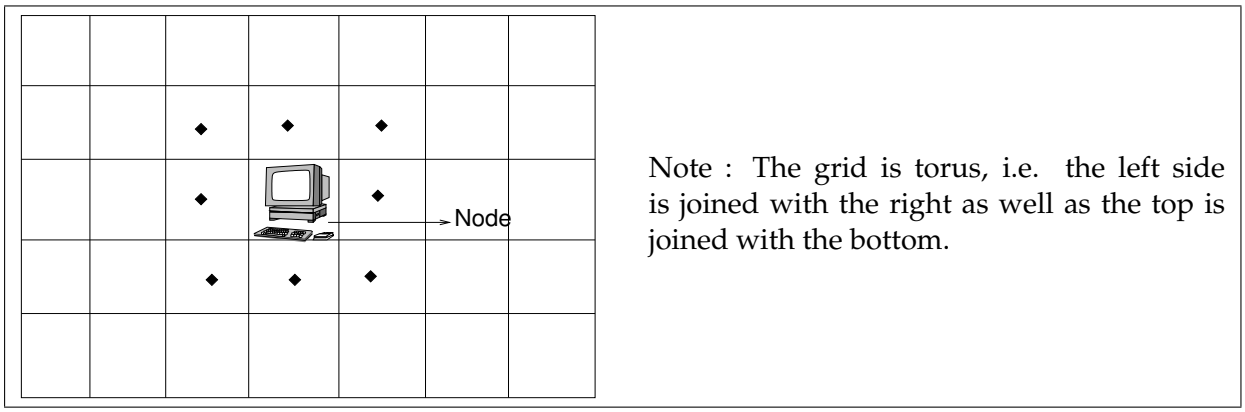


Figure 5: A node in the p2p network and its neighborhood

2.3.1 Topology and profile

We have modeled the p2p network as a toroidal grid where each node in the grid is conceived as a member of the p2p network (*Fig. 5*). Due to the grid structure, each node has a fixed set of eight neighbors, the neighbors are marked in *Fig 5*. Each node has two profiles - the *informational profile* and the *search profile*.

The Informational Profile (P_I) of the node is formed from the information which is shared by the node with other community members in the p2p network. The node's profile is created as a result of the analysis of data stored by the given node. The Search Profile (P_S) of the node is built from the informational interest of the users of a node. In general, this may differ from the information stored on the node. For initial study, we have modelled both information and search profile by one 10 bit token. The query message packet (M) is also a 10-bit token.

The affinity measurement between a profile (P) and a message packet (M) is discussed next.

2.3.2 Affinity measure

Similarities between a profile P and a query message packet (M), are measured in terms of the number of bits that are identical. That is,

$$sim(P, M) = d - HD(P, M) \tag{1}$$

where HD is the *Hamming distance* between P and M.

Since tokens are of finite length (d), there can be at most 2^d unique tokens present in the network. The method to determine the frequency of each of the unique alternatives is discussed next.

2.3.3 Distribution of profile

The profile (both information and search) is created in a realistic fashion following *Zipf's law*. Zipf's law, named after the Harvard linguistic professor George Kingsley Zipf (1902-1950), describes the observation that the frequency of occurrence of an event (here token) t, as a function

2	3	1	3
2	2	2	1
1	2	1	0
3	2	2	0

Note: The unique token set is {0, 1, 2, 3} and the total number of tokens are 16; The rank of the tokens in terms of frequency are {2,1,3,0}. Therefore from *Relation 2*,

$$\frac{\mathcal{N}}{1} + \frac{\mathcal{N}}{2} + \frac{\mathcal{N}}{3} + \frac{\mathcal{N}}{4} = 16 \Rightarrow \mathcal{N} = 7.68$$

whereby {2, 1, 3, 0} occurs { 7 ($\approx$ 7.68), 4 ($\approx \frac{7.86}{2}$), 3 ($\approx \frac{7.86}{3}$), 2 ($\approx \frac{7.86}{4}$)} times respectively.

Figure 6: Distribution of tokens in a p2p network following Zipf's law

of the rank i where the rank is determined by the above frequency of occurrence, is a power-law function $t_i = \frac{1}{i^a}$ with the exponent a close to unity [161]. That is, if the most frequent token t_1 has occurred 1000 times, t_2 occurs 500 times, t_3 occurs 333 times and so on. It is generally believed that the most of the features in P2P network follows *Zipf's law* [3].

Since in our model each token is of finite length (d), there are 2^d different possible tokens. Therefore, in our model, Zipf's law will be implemented in the following way. The total number of tokens (N) of (say) Information Profile in the whole p2p network is given by

$$N = m_1 \times m_2$$

where $m_1 \times m_2$ is the dimension of the grid and each node in a grid hosts a token as *Information Profile*.

Therefore, these N tokens can be of the total possible 2^d types, hence from *Zipf's law*, the i^{th} token will have $\frac{\mathcal{N}}{i^a}$ occurrence, where

$$\frac{\mathcal{N}}{1^a} + \frac{\mathcal{N}}{2^a} + \cdots + \frac{\mathcal{N}}{i^a} \cdots \frac{\mathcal{N}}{(2^d)^a} = N \tag{2}$$

From the equation, $\mathcal{N}$ can be calculated directly. A small example of *Zipf's law* is given in *Fig. 6*.

On the basis of the above discussed model, we now elaborate the search algorithm *ImmuneSearch*.

2.4 *ImmuneSearch*: Immune-inspired algorithm for search tasks

The *ImmuneSearch* is a distributed algorithm and will be implemented individually and independently at each node. The search algorithm consists of two parts, the line of motion followed by query message packets through the network and the topology evolution initiated due to the search.

2.4.1 Movement

The search in our p2p networks is initiated from the user node. The user (U in *Fig. 7*) emanates message packets (which is derived from the *search profile* of the user) to its neighbors and the packets are thereby forwarded to the surroundings. The method of spreading of the message packets to its neighborhood forms the basis of the algorithm. In our scheme, the spreading algorithm is inspired from immune system. The message packets are conceived as *antibody*, whereas the profile it wants to find out is conceived as *antigen*. *For ease of understanding, we hereby refer message packets as antibody.*

Similar to their behavior in natural immune systems, the antibodies undergo random walk across the grid, but when they come across a matching antigen (an antigen is the information profile of any arbitrary node), that is, the similarity between the antibody and antigen is above a threshold, it undergoes proliferation (as shown in position A of *Fig. 7*), so as to find more and more nodes with similar information profile around its neighborhood. Besides proliferation, mutation is initiated which brings in diversity in the search and helps the user node to find a wide variety of search items. (The different colored message packets around A in *Fig. 7*.) The mutation results in finding new items which may not have been initiated exactly for search but which produces surprisingly new results much to the user's delight. The mechanism is next presented in algorithmic form.

Each node executes the algorithm independently. The algorithm is initiated on a particular node as soon as an antibody (M) (query message packet) enters in that node. Therefore, the algorithm named *Movementp2p* is presented with respect to a particular node A.

Algorithm 1. Movementp2p
Input : Antibody(M)
Output : Movement and Search Results
If (sim(P_i,M) $<$ Threshold)
Perform random walk on the grid.
else
Perform proliferation and mutation.

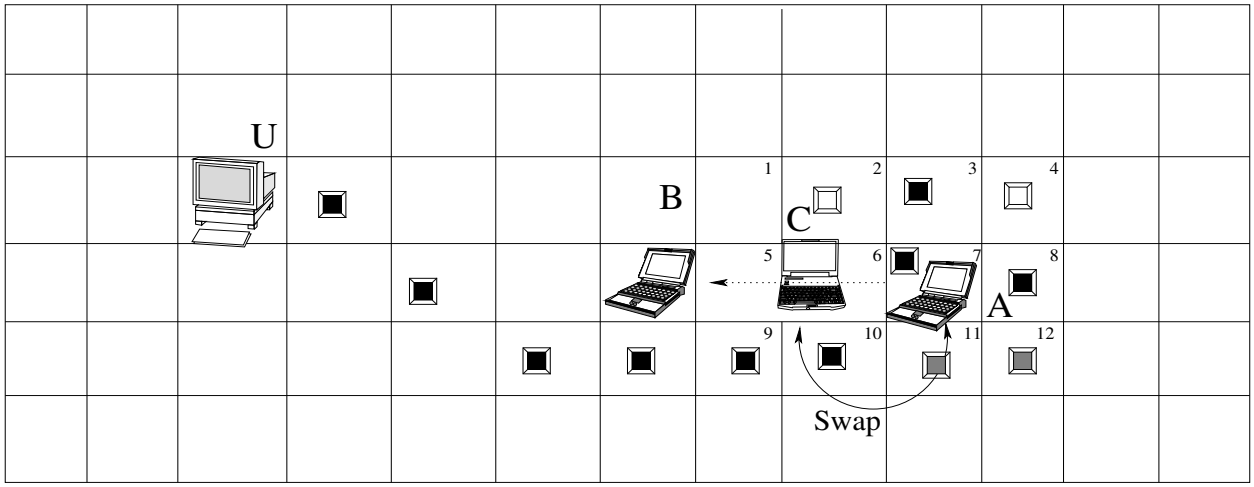
The user node which initiates the search executes the following task.

Algorithm 2. InitiateSearch
Float Antibodies(M) around the neighbors of the user node.

As is clear from the algorithm, that the effect of proliferation and mutation initiates a differential movement around the neighbors of the nodes which are already found to be similar to the antibody. This implicitly points to the fact that the topology of the network should be arranged in a fashion such that nodes which have similar profiles should come close to each other and provide a specialized environment for enhanced proliferation rate.

2.4.2 Topology evolution

Since each individual node determines which are the neighbors to join, it can change their neighborhood configuration at will. Since we are working with a fixed neighborhood condi-



Swap between C and A: C which resides in grid no. 6 has initially neighbors residing in grid numbers {1, 2, 3, 5, 7 (= A), 9, 10, 11}; while A which resides in grid no. 7 has initially neighbors residing in grid number {2, 3, 4, 6 (= C), 8, 10, 12}
After swap, A has neighbors {1, 2, 3, 5, 6 (= C), 9, 10, 11}, while C has neighbors {2, 3, 4, 7(= A), 8, 10, 11, 12}. That is, in swap, in effect, the neighbors are changed, not the physical position of the machine.

Figure 7: The Search Mechanism

tion, i.e. each node can have eight neighbors, the neighborhood change follows the method illustrated in *Fig. 7*. It is seen that the node has changed from A to B. During its movement from A to B, at each step it swaps its position that is, neighborhood information with intermediate nodes (as C in *Fig. 7*).

The topology or the neighborhood configuration of a node (say A) is evolved based on the similarity between the node and the user node (say U) which has initiated a search. The similarity can be of two types;

- Similarity between P_I (*information profile*) of the node (A) and the antibody (M) initiated by the node (U).
- Similarity between search profile (P_S) of the node A and the antibody (M) initiated by the user node U.

The amount of distance A is brought closer to the user node (U) is proportional to

- the similarity between them (A and U) either in terms of P_I or P_S.
- the distance between node U and node A.

The movement of machine is also controlled by another important concept which is borrowed from natural immune systems - *aging*. The movement of a node gets restrained as it ages. It is assumed that the more number of times a node encounters antibodies, it ages. That is, the

longer it stays in the environment (p2p network), it is assumed that the node has found its correct position and the less is its response to any call for change in neighbors towards any user node U.

Moreover, if the search profile (P_S) of a node A matches with M, the movement of that node A towards user node U is initiated only if it is younger than U. If the node A has performed search more times than U, it is assumed to be in a more stable position than U and consequently there is no further movement.

The aging concept lends stability to the system, thus a node entering the p2p network, after undergoing the initial changes in neighborhood finds the correct position. The algorithm for topology evolution is now elaborated. The *Topology Evolution* is executed on a node A, only when the similarity between antibody (M) and the profile (P) of A is above a threshold.

Algorithm 3. *Topology Evolution*
Output : Changed Position of A.
x_1 = *sim(P_I,M) /* P_I - Information Profile of A */*
x_2 = *age(A)*
x_3 = *dist(A, U)*
x_4 = *sim(P_S,M) /* P_S - Search Profile of A */*
x_5 = *age(A) - age(U)*
if ($x_1 >$ *Threshold)*
  mov(A) α $\frac{x_1 \cdot x_3}{x_2}$
if ($x_4 >$ *Threshold and* $x_5 > 0$)
  mov(A) α $\frac{x_4 \cdot x_3}{x_2}$

We are now in a position to present the main algorithm *ImmuneSearch* which is a combination of *Algorithms 1* & *3*. The ImmuneSearch is presented with respect to a particular node A. Each node applies the *ImmuneSearch* algorithm whenever it encounters an antibody.

Algorithm 4. *ImmuneSearch*
Input : Antibody (M)
Output : Movement
if (sim(P_I,M) > Threshold)
{
  Proliferate and mutate antibodies in neighborhood.
  Topology Evolution(A)
}
else
  Randomly forward the antibody M to any of the neighborhood.

The initial simulation results which establish the efficiency of the algorithm are discussed next.

2.5 Preliminary experimental results

The initial experimental results illustrates the efficiency of the algorithm. We also simulate experiments with random walk as well as proliferation The set-up parameters for the grid and the profile are elaborated next.

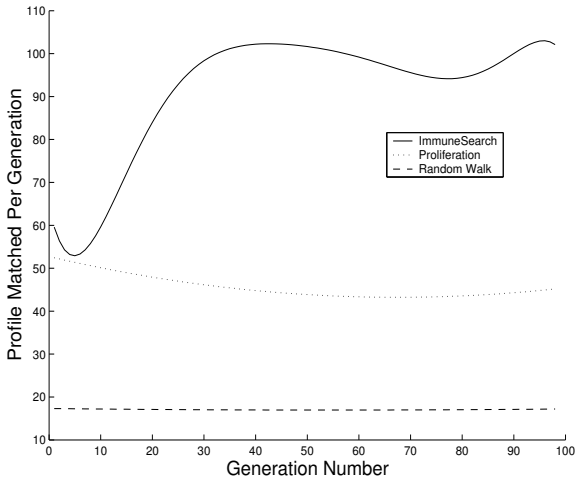


Figure 8: Efficiency of different techniques of search

2.5.1 Experimental setup

The experiment is performed on a 100×100 grid thus assuming that there are 10000 computers participating in the network. For simplicity, the dimension of both *information* and *search* profile of the node is considered to be 1, that is, both P_I & P_S comprise of a single token. Each search is initiated at a randomly chosen node and the number of search items (n_s) found within 50 time steps from the commencement of the search is calculated. This is averaged over 100 different searches whereby we obtain N_s where

$$N_s = \frac{\sum_{i=1}^{100} n_s}{100}$$

The value of N_s directly reflects the efficiency of the network. Consequently, in the graph (*Fig. 8*), each generation number actually comprises of 100 searches. The search results are represented by best fit polynomial method. This gives a clearer understanding of the trends of the results.

2.5.2 Experiments

This experiment is carried out with the assumption that no node/machine leaves the system. We have initiated three types of experiments with this model, the random movement, the proliferation and the *ImmuneSearch*. The graph of *Fig. 8* displays the performance analysis with the three different models. The x-axis of the graph shows the generation number while the y-axis

represents the average number of search items (N_s) found in the last 100 searches. It is seen that the number of search items (N_s) found is higher in proliferation than in random search. However, as expected, N_s doesn't increase over time. In the case of *ImmuneSearch*, the number of search dramatically increases by almost 100%.

2.6 Further work in progress

We have presented an initial work on developing search mechanisms on grid topology based on concepts of immune system. However, we are performing more experiments to understand the reasons for its better performance. We are also studying the effect it produces at the face of heavy unreliability of the machines participating in *p2p network*. Moreover, in the near future we have plans to test the algorithm on other more realistic type of topologies like scale-free network, random network etc. We expect, in fact, the results will be better in the realistic networks, because there will be no restriction about the number of neighbors of each node. This will add to another level of flexibility to the system. Moreover, since we don't have to maintain the neighborhood constraint as strictly as grid, the swapping of neighborhood of the peers can be done much more easily. Adding to it, the diameter of these type of networks are typically small ($\log(n)$) compared to grid ($\sqrt{n}$), generally making the search more efficient. However, all these has to be tested to provide the final comments.

Part III
Topology management

3 Membership management in overlay networks

A key aspect of a large and dynamic overlay network is membership management and the possibility of information dissemination to all members of the network. Because of the large size and dynamism of these networks, it is not possible to maintain a complete list of members at all nodes. Furthermore, a central server that provides membership information introduces scalability and reliability problems. For these reasons, a general solution that is adopted in such settings is maintaining a *partial view* of the system at each node, that is, each node will know only a limited number of peer nodes, or *neighbors*. This neighbor set can change during the lifetime of the system. The neighbor sets of the nodes define the *overlay topology*. It can be described as a graph in which there is a directed edge between nodes a and b if b is in the neighbor set of a.

This topology can also be called the "knows-about" topology referring to its definition.

In this section we define a design space for gossip-based protocols for maintaining the topology of these unstructured overlay networks. The main purpose of this work consists in supporting applications involving distributed calculation of the average (and several other statistics) of a large set of numbers (Section 5) and its application to systems monitoring (Section 6), as well as load balancing, which is discussed in Bison Deliverable D08.

This design space will include LPBCAST [55] and NEWSCAST [87] as special cases. The motivation for introducing this design space is to allow a systematic study of the different design choices. Further results on these protocols can be found in [86].

3.1 Biological analogies

Unstructured knows-about ON are analogous to social relationship networks, biological relationship networks like food chains or simply a physical proximity topology like the one defined by cells in an multicellular organism. Building this analogy further, the typical functions that can be implemented based on these topologies are *epidemic broadcasting* and *diffusion*. In this document we will focus on these two analogies.

In epidemic broadcasting the broadcasted information is analogous to an infection, and the topology is analogous to the network of contacts that can potentially transmit the infection [34]. It is well known that this paradigm is extremely efficient in fully distributed settings [56]. In our work we will apply this paradigm in several contexts like finding extremal values in a large distributed system (Section 5) or for propagating alarm signals in monitoring applications (Section 6).

Diffusion has several applications as well. In diffusion in biological or physical systems, the key idea is that there is a *source* which contains some static amount of diffusive material (or, alternatively, continuously produces such material) and this material is continuously spreading

throughout the system to the direction of lower concentration. This mechanism is believed to play an important role in biological self-organization and pattern formation.

In a computer system, the same mechanism can find several applications, as those already mentioned of distributed calculation of statistics of large set of numbers and system monitoring.

3.2 Related work

Epidemic protocols are becoming more and more popular since the publication of the seminal paper by Demers et al. [34]. A recently completed survey by Eugster et al. provides an excellent introduction to the field [56]. Epidemic algorithms have been applied to solving several practical problems like database replication [34], failure detection [152] and resource monitoring [151]. A large body of theoretical work is also available due to the general importance of understanding epidemics [5] and its close relation to random graph theory [18].

There are countless protocols for managing different kinds of topologies in an adaptive fashion. Here we focus only on those that maintain a random (or unstructured) topology. The most well-know examples are SCAMP [61], lpbcast [55], a method to build random expander graphs [100, 119].

3.3 Design space for gossip-based protocols

3.3.1 System model

The basic abstract component of the system is a *node*. We assume that we are given a set of nodes, and each node is connected to a network. Furthermore, we assume that each node has an *address* and that it is sufficient to know the address of a node to send a message to it. In other words we require a routing service. We also assume that each node has access to a clock. The clocks of the different nodes need not be synchronized. Obviously, processes with access to the core Internet satisfy these conditions.

Each node possesses a *partial view* which is a limited-size set of *node descriptors*. The maximal size of the view will be denoted by c. In the following we will simply say view instead of partial view. A node descriptor is a record which contains the address of the node and any additional information required by the specific implementation of the protocol. in our case the descriptor contains the *hop count* of the descriptor as additional information. Hop count will be defined later during the discussion of the protocol.

3.3.2 The protocol

The frame of the protocol is shown in Figure 9. The protocol is defined by an active and a passive thread. The active thread periodically contacts a peer and performs an information exchange step according to the implementation of the methods SELECTPEER and SELECTVIEW, and the setting of the two boolean parameters PUSH and PULL. We will describe these components in detail, but first let us explain the MERGE operation.

```
do forever
  wait(T time units)
  p = selectPeer()
  if push then
    // 0 is the initial hop count
    myDescriptor ← (myAddress, 0)
    buffer ← merge(view,{myDescriptor})
    send buffer to p
  if pull then
    receive view_p from p
    view_p ← increaseHopCounts(view_p)
    buffer ← merge(view_p,view)
    view ← selectView(buffer)
```

(a) active thread

```
do forever
  (p, view_p) = waitMessage()
  view_p ← increaseHopCounts(view_p)
  if pull then
    // 0 is the initial hop count
    myDescriptor ← (myAddress, 0)
    buffer ← merge(view,{myDescriptor})
    send buffer to p
  buffer ← merge(view_p,view)
  view ← selectView(buffer)
```

(b) passive thread

Figure 9: The skeleton of a gossip-based protocol for maintaining unstructured overlay networks.

The view can be described as an ordered set data structure which maintains the following invariable properties: (a) it contains at most one descriptor for each node (i.e. it is a set w.r.t. the nodes) and (b) it is ordered according to the hop count stored in the descriptors in increasing order. For example, in this sense we can meaningfully talk about the *first* or *last* k elements of the view. Method INCREASEHOPCOUNT increases the hop count stored in each descriptor in its parameter view set by one. Method MERGE returns a view which contains exactly one descriptor for each node stored in any of the two views in its parameter list. Furthermore, if a node has a descriptor in both views, it keeps the one with the minimal hop count.

Now let us turn to the implementation of the components that form the heart of the protocol.

3.3.3 Peer selection

Method SELECTPEER returns a node address stored in the current view. The implementation of this method can be based on the additional information that is stored in the descriptors, in our case, hop count.

The contract of this method includes the assumption that the peer address returned is alive. This can be achieved if method SELECTPEER performs a ping probe to the address before returning it.

It is evident that the possibilities of implementing this method are practically endless, especially if we extend the node descriptors with additional information. In this paper we will consider three simple implementations, as described bellow.

Random The returned address is determined by uniform random sampling of the view.

Head The returned address is the first accessible address stored in the view. Recall that in the view the descriptors are ordered according to increasing hop count.

Tail The returned address is the last accessible address stored in the view.

3.3.4 View selection

Method SELECTVIEW selects a subset of maximal size c from the view passed to it as parameter. As in the case of peer selection, from the huge space of possible implementations we select the following three simple algorithms.

Random The returned view is determined by taking c random samples from the parameter view without replacement.

Head The returned view is defined by the first c elements in the parameter view. Recall that in the view the descriptors are ordered according to increasing hop count.

Tail The returned view is defined by the last c elements in the parameter view.

3.3.5 Symmetry of communication

The symmetry/assymmetry of the communication is defined by the boolean parameters PUSH and PULL. Clearly, at least one of them must be set to true, otherwise nothing happens. In this paper we will consider all the remaining three possible settings.

3.3.6 Our protocol design space

Taking the implementations described above into account, we can now define the protocol design space. This space is defined by all possible combinations of the components according to the Cartesian product

$$\{\text{rand,tail,head}\} \times \{\text{rand,tail,head}\} \times \{\text{push,pull,pushpull}\}$$

where the first set refers to the peer selection method, the second set to the view selection method and the last set to the adopted symmetry model. According to this formulation, a 3-tuple will uniquely define a protocol. For example, (rand,rand,push) is LPBCAST [55], (rand, head, pushpull) is NEWSCAST [87].

4 Minimum power connectivity in wireless networks

Among the most crucial issues related to mobile ad-hoc and sensor networks is that of operation in limited energy environments, since devices are usually equipped with battery with a limited lifetime.

Since radio signals have non-linear attenuation properties, it is very energy-consuming to transmit a signal far away. Another drawback of long-distance transmissions is that they tend to produce noise over the network, and for this reason they should be avoided.

The previous issues can be seen as correlated, and they can be handled together by taking advantage of the so-called *wireless multicast advantage* property (see, for example, Wieselthier et al. [158]). This property is based on the observation that in wireless networks, devices are usually equipped with omnidirectional antennae, and for this reason multiple nodes can be reached by a single transmission. In the simple example of Figure 10, where transmission powers p are depicted, nodes j and k receive the signal originated from i and directed to m because j and k are closer to i than m, i.e. they are within the transmission range of a communication from i to m.

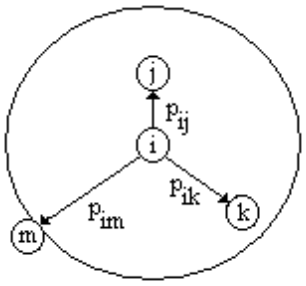


Figure 10: Minimum power connectivity problem. Communication model.

The property above can be used to minimize the total transmission power required to connect all the nodes. This produces a network where the sum of devices' lifetimes is maximized and, as a side effect, short distance transmissions are preferred. In particular, a well-known crucial topology problem, the *minimum power connectivity problem* (sometimes also referred to as *minimum power broadcast problem*) arises.

For a given set of nodes of a wireless network, the *minimum power connectivity problem* is to assign transmission powers to the nodes of the network in such a way that the network is connected (eventually strongly connected) and the total power consumption is minimized. The common assumption that no power expenditure is involved in reception/processing activities will be adopted.

4.1 Related work

Most of the work presented in recent years is focalized on the development of centralized techniques, where the complete knowledge of pair-wise distances between the nodes is assumed. Often the extra assumption that a communication link is established only if both nodes have transmission range at least as big as the distance between them. This last assumption is sometimes justified by technical reasons (see Althaus et al. [4]), and then it can be adopted or not depending on the hardware used and on the target application of the network. We will consider mainly the bidirectional case, but our study will cover also the less constrained case where bidirectional links are not required.

Some mixed integer programming formulations which can be adapted to the static, centralized version of the problem, are presented in Das et al. [29], unfortunately without any experimental result. A branch and cut algorithm based on another new integer programming formulation is proposed in Althaus et al. [4]. In Montemanni and Gambardella [115] one more integer programming formulation and some new valid reinforcing inequalities for this formulation are presented.

Heuristic approaches to the problem can be derived from those proposed in Wieselthier et al. [158], Marks II et al. [85] and Das et al. [28], where some constructing algorithms, an evolutionary approach using genetic algorithms and an ant colony system approach are respectively proposed. Some of these heuristic have been also tested in distributed environments, but rarely dynamism has been taken into account.

4.2 Complex adaptive system framework and innovative aspects

It is natural to frame the minimum power connectivity problem in terms of complex adaptive systems, since in the problem every device has to operate in behalf of the whole network, while it is very likely that it does not even know the characteristics of the full network.

In practice, every device has to regulate its own power in order to minimize the total power consumption over the network in such a way to maintain the network connected, and has to do this while it only has a partial vision of the problem.

Techniques inspired by biology may than apply particularly well for the given problem.

We are going to develop new algorithms for the static, centralized problem and we are going to evaluate them into a distributed, dynamic environment. This appears to be the direction of the research currently going on in the field.

4.3 General approaches for centralized and distributed strategies

According to the research currently going on, we have started by implementing solutions for the centralized problem without dynamism. In particular we have developed methods based on new Integer Programming formulations for the problem. They are able to provide exact solutions to the problem.

Unfortunately, as it was expected, these exact techniques seems to be quite expensive in terms of computational complexity, so we plan to develop some biology-inspired/meta-heuristic

methods (e.g. Simulated Annealing - see Kirkpatrick et al. [95], Ant Colony Systems - see Dorigo et al. [43]) which should be able to improve the quality of the solutions provided by the local search approaches currently available, while ensuring short computation times.

The quality of the solution provided by heuristic algorithms will be evaluated by comparing them to those produced by the exact algorithms, and this should permit a precise performance evaluation.

The next step of our research will be to move into a distributed environment, where the performance of the centralized algorithms previously developed will be tested under different situations. In particular we plan to run experiments where different levels of knowledge of the whole network are considered, in order to individualize a trade-off between the quality of the solutions and the overhead due to the exchange of network information among nodes. New ad-hoc algorithms for the distributed environment may be developed at this point.

A farther step will be to take into account dynamism and to evaluate how the distributed algorithms (considering here also those derived from centralized methods) react when different levels of dynamism are considered. Changes/improvements to the algorithms may be introduced at this point.

4.4 Integer programming formulation for the centralized problem

In this section we briefly summarize some preliminary results we have obtained on the (strongly connected version of the) centralized problem. After a brief description of the problem, we present a new integer programming formulation and some preliminary computational experiments.

4.4.1 Problem description

In order to formalize the problem, a model for signal propagation has to be selected. We adopt the model presented in Rappaport [124]. Signal power falls as $\frac{1}{d^\kappa}$, where d is the distance from the transmitter to the receiver and κ is an environment-dependent coefficient, typically between 2 and 4 (we will set $\kappa = 4$). Under this model, and adopting the usual convention (see, for example, Althaus et al. [4]) that every node has the same transmission efficiency and the same detection sensitivity threshold, the power requirement for supporting a link from node i to node j, separated by a distance d_{ij}, is then given by

$$p_{ij} = (d_{ij})^\kappa \tag{3}$$

Using the model described above, power requirements are symmetric, i.e. $p_{ij} = p_{ji}$.

MPB can be formally described as follows:
Given the set V of the nodes of the network, a *range assignment* is a function $r : V \to \mathcal{R}^+$. A *bidirectional link* between nodes i and j is said to be established under the range assignment r if $r(i) \geq p_{ij}$ and $r(j) \geq p_{ij}$. Let now $B(r)$ denote the set of all bidirectional links established under the range assignment r. MPB is the problem of finding a range assignment r minimizing $\sum_{i \in V} r(i)$, subject to the constraint that the graph $(V, B(r))$ is connected.

As suggested in Althaus et al. [4], a graph theoretical description of MPB can be given as follows:
Let $G = (V, E, p)$ be an edge-weighted graph, where V is the set of vertices corresponding to the set of nodes of the network and E is the set of edges containing all the possible (unsorted) pairs $\{i, j\}$, with $i, j \in V$, $i \neq j$. A cost p_{ij} is associated with each edge $\{i, j\}$. It corresponds to the power requirement defined by equation (3).

For a node i and a spanning tree T of G (see, for example, Kruskal [98]), let $\{i, i_T\}$ be the maximum cost edge incident to i in T, i.e. $\{i, i_T\} \in T$ and $p_{ii_T} \geq p_{ij}\ \forall \{i, j\} \in T$. The *power cost* of a spanning tree T is then $c(T) = \sum_{i \in V} p_{ii_T}$. Since any connected graph contains a spanning tree, and a broadcast tree must be connected, MPB can be described as the problem of finding the spanning tree T with minimum power cost $c(T)$. This observation is at the basis of the integer programming formulation which will be presented in Section 4.4.2.

4.4.2 Mixed integer programming formulation

The mixed integer programming formulation described in this section can be seen as an evolution of one of those proposed in [29]. It is based on the representation of the spanning tree problem through a network flow model (see Magnanti and Wolsey [107]).

A weighted, directed, complete graph $G' = (V, A, p)$ is derived from G by defining $A = \{(i, j) |\ i, j \in V\}$, i.e. for each edge in E there are the respective two arcs in A, and a dummy arc (i, i) with $p_{ii} = 0$ is inserted for each $i \in V$. p_{ij} is defined by equation (3) when $i \neq j$.

In order to describe the formulation, we also need the following definition.

Definition 1. *Given $(i, j) \in A$, we define the* ancestor *of (i, j) as*

$$a_j^i = \begin{cases} i & \text{if } p_{ij} = \min_{k \in V}\{p_{ik}\} \\ \arg\max_{k \in V}\{p_{ik} | p_{ik} < p_{ij}\} & \text{otherwise} \end{cases} \tag{4}$$

According to this definition, (i, a_j^i) is the arc originated in node i with the highest cost such that $p_{ia_j^i} < p_{ij}$. In case an *ancestor* does not exist for arc (i, j), vertex i is returned, i.e. the dummy arc (i, i) is addressed.

In the formulation IP, an arbitrary node s, the one from which to broadcast, is elected the root of the spanning tree, and one unit of flow is sent from s to every other node. Variable z_{ij} represents the flow on arc (i, j). Variable y_{ij} is 1 when node i has a transmission power which allows it to reach node j; $y_{ij} = 0$ otherwise.

$$(IP)\ \text{Min} \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{5}$$

$$\text{s.t. } y_{ij} \leq y_{ia_j^i} \qquad \forall (i,j) \in A, a_j^i \neq i \tag{6}$$

$$z_{ij} \leq (|V|-1) y_{ij} \qquad \forall (i,j) \in A \tag{7}$$

$$z_{ij} \leq (|V|-1) y_{ji} \qquad \forall (i,j) \in A \tag{8}$$

$$\sum_{(s,j)\in A} z_{sj} - \sum_{(k,s)\in A} z_{ks} = |V| - 1 \tag{9}$$

$$\sum_{(i,j)\in A} z_{ij} - \sum_{(k,i)\in A} z_{ki} = -1 \qquad \forall i \in V \backslash \{s\} \tag{10}$$

$$z_{ij} \in \mathcal{R} \qquad \forall (i,j) \in A \tag{11}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{12}$$

In formulation IP an incremental mechanism is established over y variables (i.e. transmission powers). The costs associated with y variable in the objective function (5) are then given by the following formula:

$$c_{ij} = p_{ij} - p_{ia_j^i} \ \ \forall (i,j) \in A \tag{13}$$

c_{ij} is then equal to the power required to establish a transmission from nodes i to node j (p_{ij}) minus the power required by nodes i to reach node a_j^i ($p_{ia_j^i}$). In Figure 11 the costs arising from the example of Figure 10 are depicted.

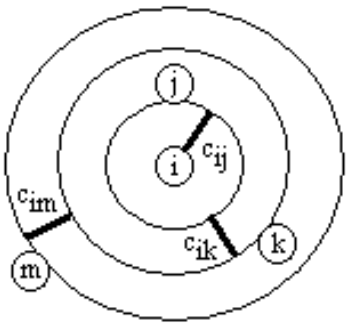


Figure 11: Minimum power connectivity problem. Costs for the mathematical formulation IP.

Constraints (6) realize the incremental mechanism by forcing the variables associated with arc (i, a_j^i) to assume value 1 when the variable associated with arc (i,j) has value 1, i.e. the arcs originated in the same node are activated in increasing order of p. Inequalities (7) and (8) connect the flow variables z to y variables. Equations (9) and (10) define the flow problem, while (11)s and (12)s are domain definition constraints. The interested reader can find a more detailed description of the flow problem behind the formulation above in [107].

Some new valid inequalities for formulation IP are being developed. These inequalities are able to speed up the solving time for the formulation. In Section 4.4.3 some preliminary computational experiments show that formulation IP reinforced with these new inequalities under development is faster than a method recently appeared in the literature.

4.4.3 Preliminary computational results

The tests presented in this section are on problems randomly generated as described in [4]. For each problem of size $|V|$ generated, $|V|$ points (nodes) have been chosen uniformly at random from a grid of size 10000×10000.

Tests have been carried out on a SUNW Ultra-30 machine, and ILOG CPLEX 6.0[11] has been used to solve integer programs.

In Table 1 we present the average computation times (over 50 runs) for different values of $|V|$. We report the times (in seconds) required by the approach described in [4] and to solve the integer program described in Section 4.4.2 reinforced with some new valid inequalities we are developing.

Since we carried out our tests on a SUNW Ultra-30 machine, and we want to compare our results with those reported in [4], which are obtained on an AMD Duron 600MHz PC, we divided our computation times by a factor of 3.2, as suggested in Dongarra [41]. This makes the computation times comparable.

Table 1: Minimum power connectivity problem. Exact methods comparison. Averages over 50 runs.

$\|V\|$	Computation time (sec)	
	Althaus et al. [4]	IP
10	0.67	0.06
15	5.68	0.23
20	22.20	2.68
25	58.90	10.36
30	201.00	69.19
35	712.00	389.47
40	4725.00	3089.40

Computation times reported in Table 1 indicate that the exact algorithm we suggest, i.e. solving the integer programming formulation IP reinforced with the inequalities we are developing, clearly outperforms the (more complex) method described in [4], especially for small and medium size problems.

[11] `http://www.cplex.com`

Part IV
Collective computations

5 Aggregation in overlay networks

5.1 Introduction

A useful functional building block for monitoring and control is *aggregation* as we already mentioned in Section 3. Aggregation is the collective name of several functions that provide a variety of statistical information about a system [150]. These functions include finding extremal values of some property, counting, computing averages and sums, etc. Aggregation can provide users or participants of a P2P network with important information like the number of nodes connected to the network or the total amount of free space in a distributed storage. Furthermore, aggregation can be used as a building block for more complex protocols. For example, the knowledge of the average load in a distributed network can be exploited to implement near-optimal load-balancing schemes [89].

Aggregation protocols can be divided in two categories: *reactive* and *proactive*. Reactive protocols respond to specific queries issued by nodes in the network. The answers are returned directly to the sender of the query [73, 11]. Proactive protocols, on the other hand, continuously provide the value of some aggregate to *all* nodes in the system, in an *adaptive* fashion. Adaptivity means that if the aggregate changes due to network dynamism or because of variations in the values to be aggregated, the output of the aggregation protocol should follow this change reasonably quickly. Proactive protocols are often useful when aggregation is used as a building block for completely decentralized protocols. For example, in the load-balancing scheme cited above, the knowledge of the average load is used by each node to decide when it can stop transferring load [89].

In this section we introduce a robust and adaptive protocol for calculating aggregates in a proactive manner. The core of the protocol is a simple epidemic-style scheme [88] in which aggregation is done in the style of an anti-entropy epidemic protocol, typically used for updating distributed databases. Periodically, each node selects a random peer and communicates with it to bring their states up-to-date [34]. The difference is that instead of resolving differences between databases, the elementary step consists of some computation based on the values maintained by the two communicating peers. Further results on our approach on aggregation can be found in [116].

5.2 System model

We consider a P2P network consisting of a large collection of *nodes* that communicate through the exchange of messages and are assigned unique identifiers. We assume that nodes are connected through an existing routed network, such as the Internet, where every node can potentially communicate with every other node. To actually communicate, each node has to know the identifiers of a set of other nodes (its *neighbors*). This neighborhood relation over the nodes

defines the topology of the *overlay network*. Given the large scale and the dynamicity of our envisioned system, neighborhoods are normally limited to a rather small subsets of the entire network.

5.3 The basic idea

do forever
wait(δ time units)
$q \leftarrow$ GETNEIGHBOR()
send s_p to q
$s_q \leftarrow$ receive(q)
$s_p \leftarrow$ UPDATE(s_p, s_q)

(a) active thread

do forever
$s_q \leftarrow$ receive(*)
send s_p to sender(s_q)
$s_p \leftarrow$ UPDATE(s_p, s_q)

(b) passive thread

Figure 12: Skeleton of the protocol executed by node p.

Each node in the network holds a numeric value. In a practical setting, this value can characterize any (dynamic) aspect of the node or its environment (e.g., load in a distributed computing application, or temperature monitored by a sensor network). The task of a proactive protocol is to continously provide all nodes with an up-to-date estimate of the aggregate function, computed over the set of values held by all nodes

Our basic aggregation protocol is based on the push-pull epidemic-style scheme illustrated in Figure 12. Each node p executes two different threads. The *active* one periodically initiates an *information exchange* with a peer node q selected randomly among its neighbors, by sending a message containing the local state s_p and waiting for a response from q containing its state s_q. The *passive* thread waits for messages sent by an initiator and replies with its local state. The term push-pull refers to the fact that each information exchange is performed in a symmetric manner: both peers send and receive their states.

The period of wall clock time between two consecutive information exchanges is called the *cycle length*, and is denoted by δ. Even though the system is not synchronous, it is often convenient to talk about *cycles* of the protocol. Cycles are consecutive wall clock intervals of length δ counted from some convenient starting point.

Method UPDATE builds a new local state based on the previous local state and the state received during the information exchange. The output of UPDATE depends on the specific function implemented by the protocol. Here, we limit the discussion to AVERAGE. Additional functions are described in Section 5.4.

To implement AVERAGE, each node stores a single numeric value representing the estimate of the aggregation output. Each node initializes the estimate with its local value. Method UPDATE(s_p, s_q), where s_p and s_q are the estimates exchanged by p and q, returns $(s_p + s_q)/2$. After one exchange, the sum of the two local values does not change, since they have just balanced their values. So, the operation does not change the global average either; it only decreases the variance over all the estimates in the system.

It is easy to see that the value at each node will converge to the true global average, as long as the underlying overlay is connected. In our previous work [88], we presented analytical results about the convergence speed of the averaging process. Let μ_i be the empirical mean and σ_i^2 be the empirical variance in cycle i,

$$\mu_i = \frac{1}{N}\sum_{k=1}^{N} a_{i,k}, \quad \sigma_i^2 = \frac{1}{N-1}\sum_{k=1}^{N}(a_{i,k} - \mu_i)^2 \tag{14}$$

where $a_{i,k}$ is the value maintained at node $k = 1, \ldots N$ during cycle i and N is the number of nodes in the system.

The *convergence factor* ρ_i, with $i > 1$, characterizes the speed of convergence and is defined as follows:

$$\rho_i = \frac{E(\sigma_i^2)}{E(\sigma_{i-1}^2)}. \tag{15}$$

If the overlay topology is not only connected, but also sufficiently random, it is possible to show that $\rho_i \approx 1/(2\sqrt{e})$ for $i \geq 1$. From this result, it is clear that convergence speed is exponential and so convergence can be achieved with very high precision in only a few cycles, irrespective of the network size, which confirms extreme scalability.

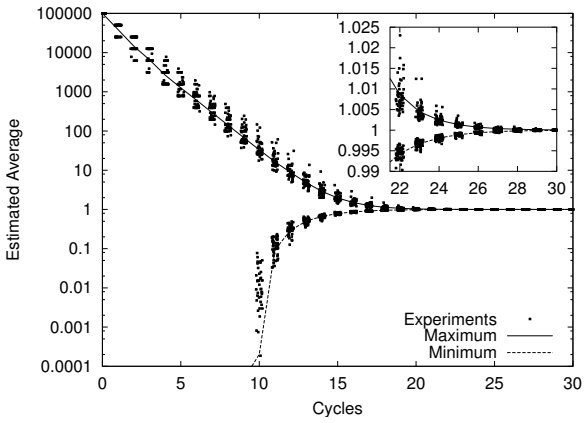


Figure 13: The behavior of the algorithm over a period of 30 cycles.

Figure 13 illustrates the behavior of the protocol. The AVERAGE protocol was run on a simulated network composed of 10^5 nodes connected through a regular random overlay network, where each node knows exactly 20 neighbors. Initially, a single node has the value 10^5, while the others have zero (so that the global average is 1). We are interested in this *peak distribution* for two reasons: first, it will be the basis of the COUNT protocol presented in Section 5.4; and second, it is the most appropriate scenario for testing robustness, as the unique peak value constitutes a single point of failure.

Results for the first 30 cycles are shown. The two curves represent the minimum and the maximum estimates of the average over all the nodes after the completion of each cycle. The curves have been obtained by averaging 50 independent experiments, shown as separate points in the figure.

5.4 A practical protocol

Building on the simple idea presented in the previous section we provide a full-fledged solution for proactive aggregation in a practical setting. Furthermore, we show how our protocol can be extended to compute other aggregate functions, including maximum/minimum, sum, product, variance and network size estimation.

5.4.1 Automatic restarting

The generic protocol described so far is not adaptive, as the output of aggregation does not take into account the dynamicity of the network and the variability of values. To provide up-to-date estimates, the protocol must be periodically *restarted*: at each node, the protocol is terminated and the current estimate is returned as aggregation output; then, the current values are used to re-initialize the estimates and aggregation starts again with fresh values.

To implement termination, we adopt a very simple mechanism: each node executes the protocol for a predefined number of cycles γ, depending on the required accuracy of the output and the convergence factor that can be obtained in the particular overlay topology adopted.

To implement restarting, we divide the execution of the protocol in consecutive *epochs* of length Δ and start a new instance of the protocol in each epoch. Depending on the ratio between Δ an $\gamma\delta$, it is possible that different epochs of the protocol are executed concurrently in the network. Thus, messages exchanged for a particular epoch have to be tagged with unique epoch identifiers.

5.4.2 Dynamic membership

When a node joins the network, it contacts a node that is already participating in the aggregation protocol. Here, we assume the existence of an out-of-band mechanism to discover such a node, and the problem of filling the neighbor set of the new node is discussed in Section 5.4.4.

The existing node provides the new node with the next epoch identifier and information about the time at which that epoch will start. Joining nodes are not allowed to participate in the current epoch; this is necessary to make sure that each epoch converges to the correct average present *at the start* of the epoch.

5.4.3 Synchronization

The protocol described so far is based on a strong assumption, i.e. the synchrony of cycles and epochs, a condition that cannot be satisfied in a large-scale distributed system; due to message delays and the potential drift between local clocks, nodes participating in an epoch cannot be perfectly synchronized.

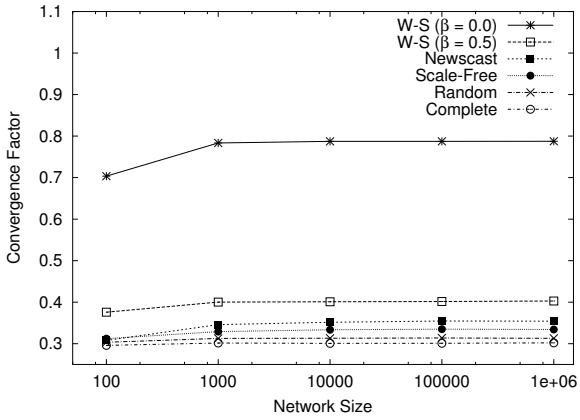
Given an epoch j, let T_j be the time interval between the first and the last nodes that started participating in epoch j. In our protocol, in the absence of any additional mechanism, the length of this interval would increase without bound. This is partly due to the drift rate of

clocks, but mainly because nodes joining the network may accumulate timing errors due to the delay of messages containing the next epoch identifier.

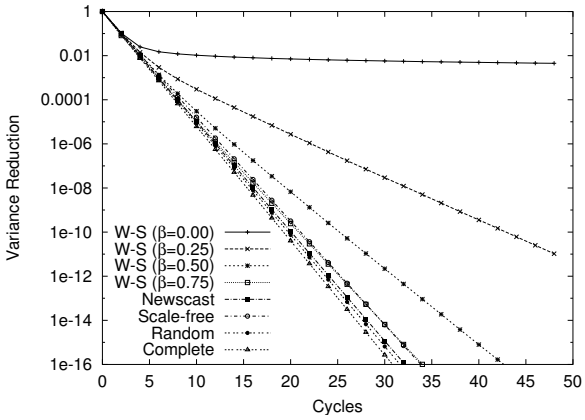
To avoid this, we modify our protocol in the following way. When a node receives an exchange message tagged with an epoch identifier larger than its current epoch identifier, it starts to participate in the new epoch and stops participating in its current epoch. In this way, based on the exponential convergence speed of epidemic broadcast protocols and the bounded delay assumption, it is possible to bound the time interval T_j for each epoch j. Note that the optimal length of the epoch (Δ) should be significantly larger that T_j. This condition holds if the average message delay is significantly shorter than Δ, since the most significant source of error is the time which is necessary to inform joining nodes about the start of the next epoch, which involves one message exchange.

5.4.4 Overlay topology for aggregation

In Section 3 we have already discussed a general set of protocols for topology management. Those protocols were based on the same epidemic-style communication model that is applied by our aggrageation protocol. Here we analyse the effect of topology on aggregation. As a dynamic topology management protocol, we will focus on NEWSCAST, that is, (rand,head,pushpull).



(a) Average convergence factor computed over a period of 20 cycles in networks composed by a variable number of nodes. Different topologies are shown; W-S corresponds to the Watt-Strogatz model.



(b) Variance reduction (normalized by the initial variance) for a network of 10^5 nodes. Different topologies are shown; W-S corresponds to the Watt-Strogatz model.

Figure 14: Behavior of the AVERAGE protocol.

The theoretical results mentioned in Section 5.3 are based on the assumption that the underlying overlay is "sufficiently random". In a more exact formulation, this means that the neighbor selected by a node when initiating communication has to be a uniform random sample of the peers. However, our aggregation scheme can be applied to generic connected topologies. To illustrate the deviation from the theoretically predicted behavior, Figure 14(a) illustrates the performance of aggregation for different topologies, by showing the average convergence fac-

tor over a period of 20 cycles, for network sizes ranging from 10^2 to 10^6. Figure 14(b) provides additional details. Here, the network size is fixed at 10^5 nodes. Instead of showing the average convergence factor, the curves represent the actual variance reduction for the same set of topologies. The reduction is normalized with respect to the initial variance. Even before describing the topologies, we can conclude that performance is independent of network size for all topologies, while it changes considerably through the different topologies. Furthermore, the convergence factor is constant through the sequence of cycle, with the only exception of topologies not sufficiently random.

All the topologies—except NEWSCAST—are static, that is, the neighbor set of each node is fixed. While in the presence of joining and crashing nodes a static topology is unrealistic, we still consider them (only in this section) due to their theoretical importance and the fact that our protocol can in fact be applied in static networks as well, although they are not in the focus of the present discussion.

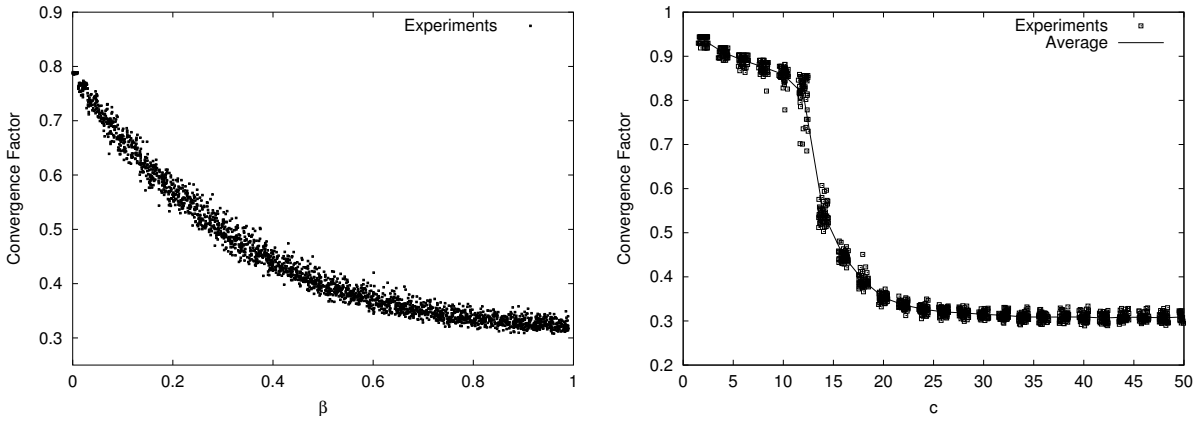
Static topologies All topologies considered have a regular degree of 20 neighbors, with the exception of the complete network (where each node knows every other node) and the Barabasi-Albert scale-free network (where the degree distribution is a power-law). For the random network the neighbor set of each node is filled with a random sample of the peers.

The remaining topologies are realistic *small-world* topologies that are often used to model different natural and artificial phenomena [8, 154]. The first class of these topologies (the Watts-Strogatz model [155]) is built starting from a regular ring lattice. The ring lattice is defined by connecting the nodes in a ring and subsequently connecting all the nodes to their nearest nodes in the ring until the desired node degree is reached. In this ring, each edge is randomly *rewired* with probabilityβ. Rewiring an edge connecting a node n to another node means to remove that edge and to add a new edge connecting n to a node picked at random. When $\beta = 0$, the ring lattice remains unchanged, while when $\beta = 1$, all edges are rewired, generating a random graph.

Figure 15(a) focuses on the Watts-Strogatz model showing the convergence factor as a function of β ranging from 0 (complete order) to 1 (complete disorder). We can observe, that although we cannot see a sharp phase transition, the increased randomness results in a better convergence factor.

Scale-free topologies form the other class of realistic small world topologies. In particular, the WWW, the Internet and P2P networks like Gnutella [127] have been shown to have a power-law degree distribution. We have tested our protocol over scale-free graphs generated using the preferential attachment method of Barabasi and Albert [9]. The basic idea of preferential attachment is that we build the graph by adding new nodes one-by-one, wiring the new node to an existing node already in the network. This existing contact node is picked randomly with a probability proportional to its degree (number of neighbors). The results are encouraging, as the observed convergence factor are similar to those obtained in random graphs.

Dynamic topologies Looking at the results above, it is clear that the topology of the overlay must be as random as possible; furthermore, in dynamic systems it must be maintained



(a) Convergence factor for Watts-Strogatz graphs with a variable β parameter.

(b) Convergence factor for NEWSCAST graphs with a variable c parameter.

Figure 15: Behavior of the AVERAGE protocol in Watts-Strogatz and NEWSCAST graphs.

continuously. The protocols discussed in Section 3 fulfill these requirements. From this set of protocols, we adopt (rand,head,pushpull), that is, NEWSCAST [87].

Nodes belonging to the network continuously inject their descriptors in the network with hop count zero, so old descriptors are gradually and automatically removed from the system and get replaced by new information. This feature allows the protocol to "repair" the overlay topology by forgetting information about crashed neighbors, which by definition are not longer active and do not inject their descriptors.

The resulting topology has a very low diameter (each node is accessible from each other node via very few links) [87]. Figure 15(b) show the performance of aggregation over a NEWSCAST network of 10^5 nodes, with c varying between 2 and 50. According to our experimental results, choosing $c = 30$ is already sufficient to obtain a fast convergence for aggregation. Furthermore, the same value is sufficient for very stable and robust connectivity.

5.4.5 Cost analysis

The communication cost and time complexity of our algorithms both follow from the properties of the aggregation protocol and are inversely related. The cycle length, δ defines the time complexity of convergence. Choosing a short δ will result in proportionally quicker convergence but higher communication costs per unit time. It is possible to show that if the overlay is sufficiently random, every Δt time units, the number of exchanges for each node can be described by the random value $1 + \phi$ where ϕ has a Poisson distribution with parameter 1. This means that on average, there are two exchanges per node (one initiated and one coming from another node), with a very low variance.

Parameter δ must be selected appropriately, in order to guarantee that with very high probability, each node will be able to safely complete the expected number of exchanges before

the next cycle starts. Failing to satisfy this requirement results in a violation of our theoretical assumptions.

Similarly, parameter γ must be chosen appropriately, based on the desired accuracy of the estimate and the convergence factor ρ characterizing the overlay network. After γ cycles, we have

$$\frac{E(\sigma_\gamma^2)}{E(\sigma_0^2)} = \rho^\gamma \tag{16}$$

where $E(\sigma_0^2)$ is the expected variance of the initial values. If ϵ is the desired accuracy of the final estimate, then $\gamma = \log_\rho \epsilon$. Note that ρ is independent of N, so the time complexity of reaching a given precision is $O(1)$.

5.5 Example aggregation functions

MIN and MAX: To obtain the maximum or minimum value among those maintained by nodes, method UPDATE(a, b) of the generic scheme of Figure 12 must return $\max(a, b)$ or $\min(a, b)$, respectively. In this case, the global maximum or minimum value will be effectively broadcast like an epidemic. Existing results about epidemic-style broadcasting [34] are applicable.

COUNT: We base this protocol on the observation that if the initial distribution is such that exactly one node has a value 1 and all the others have 0, then the average is exactly $1/N$ and N can be calculated directly.

However, achieving this property in distributed system where nodes appear and disappear continuously may not be possible. An alternative approach consists in enabling multiple nodes to start concurrent instances of the averaging protocol. Each concurrent instance is lead by a different node. Messages and data related to an instance are tagged with a unique identifier (e.g., the address of the leader). Each node maintains a map M associating a leader id with an average estimate. When nodes n_i and n_j maintaining the maps M_i and M_j perform an exchange, the new map M (to be installed at both nodes) is obtained by merging M_i and M_j in the following way:

$$\begin{aligned} M \quad = \quad & \{(l, e/2) \mid e = M_i(l) \in M_i \wedge l \notin D(M_j)\} \cup \\ & \{(l, e/2) \mid e = M_j(l) \in M_j \wedge l \notin D(M_i)\} \cup \\ & \{(l, (e_i + e_j)/2 \mid e_i = M_i(l) \wedge e_j = M_j(l)\}, \end{aligned}$$

where $D(M)$ corresponds to the domain (key set) of map M and e_i is the current estimate of node n_i.

In other words, if the average estimate for a certain leader is known to only one node, the other node is considered as having an estimate of 0.

Maps are initialized in the following way: if node n_l is a leader, the map is equal to $\{(l, 1)\}$, otherwise the map is empty. All nodes participate in the protocol described in the previous section. In other words, even nodes with an empty map perform random exchanges. Otherwise, an approach where only nodes with a non-empty set perform exchanges would be less effective in the initial phase while few nodes have non-empty maps.

Clearly, the number of concurrent protocols in execution must be bounded, to limit the communication cost involved. A simple mechanism that we adopt is the following. At the beginning of each epoch, each node may become leader of a run of the aggregation protocol with probability P_{lead}. At each epoch, we set $P_{\text{lead}} = C/\hat{N}$, where C is the desired number of concurrent runs and $\hat{N}$ is the estimate obtained in the previous epoch. If the systems size does not change dramatically within one epoch then this solution ensures that the number of concurrently running protocols will be approximately Poisson distributed with the parameter C.

SUM: Two concurrent aggregation protocols are run, one to estimate the size of the network, the other to estimate the average of the values to be summed. Size and average are multiplied in order to obtain an estimate of the sum of the values.

GEOMETRICMEAN and PRODUCT: In order to compute the geometric mean and the product of the values contained in the network, the same approach to compute the arithmetic mean and the sum may be used. Instead of returning $(a + b)/2$, method UPDATE(a, b) returns $\sqrt{ab}$. After one exchange, the product of the two local values does not change, but the variance over the set of values is decreased; the values converge toward the geometric mean. As before, once the geometric mean is known with sufficient precision, the result of a concurrent COUNT protocol may be used to obtain the product.

VARIANCE: The average of the values $\overline{a}$ and the average of the squares of the values $\overline{a^2}$ can be concurrently calculated using the averaging protocol. Then, the difference $\overline{a^2} - \overline{a}^2$ gives the estimation of the variance.

5.6 Related work

The field of distributed computation of aggregates is less established than epidemic protocols. An overview of the problem can be found in [150].

A prominent approach is Astrolabe [151] which is a hierarchical architecture for aggregation in large distributed systems. Our approach is substantially different in that it is extremely simple, lightweight, and aimed at unstructured, highly dynamic environments. In the case of our protocol the overhead of installation and maintenance is virtually negligible. A related work is [73] which is also based on a hierarchical approach. While building hierarchies indeed reduces the cost of finding the aggregates, it introduces additional overhead having to maintain this hierarchical topology in a dynamic distributed environment. Moreover, due to being hierarchical, it also needs extra effort and protocols to broadcast the result continuously over the network if all nodes need to know the result continuously.

Another recent work [11] discusses many approaches, based on spanning tree induction and using other, more redundant topologies. While being the closest to our approach, a main difference is that the protocols described there are *reactive*: aggregation is initialized from a certain point and the result is known by only that node. This makes it hard to adopt for solving our present research problem, continuous network monitoring, similarly to the other approaches mentioned above.

Part V
Monitoring functions

6 Monitoring in overlay networks

So far we have introduced dynamic unstructured topologies in Section 3 and simple algorithms for calculating the average and finding the maximum in Section 5. Now we will demonstrate the applicability of these protocols for monitoring network size, measuring the total amount of resources, or distributing alarm signals. An extended discussion of these ideas can be found in [87]. A part of the credit for the content of this section goes also to Wojtek Kowalczyk and Maarten van Steen who do not work for the BISON project.

6.1 Related work

Network and systems monitoring has since long been part of management architectures that tend to be large, complex, and difficult to scale across wide-area systems (see, for example, [77]). Recently, new insights have led to completely decentralized monitoring solutions, often involving mobile agents [104]. Also the research into scalable event-notification systems that deploy peer-to-peer technology is highly relevant to our work.

Siena is arguably one of the first large-scale event-notification systems, which effectively applies a combination of multicasting and content-based routing to efficiently notify events to interested parties [23]. However, approaches such as followed in Siena require that a network of servers is first installed before application-level multicasting can be deployed.

In this respect, more interesting is Scribe [24]. Scribe is an application-level multicasting system that is built on top of Pastry, a structured peer-to-peer network [129]. Scribe allows the formation of topic-based publish/subscribe groups in a fully dynamic and decentralized fashion.

A step further in this direction is taken in PeerCQ [62]. It is a general-purpose information monitoring system in which a client can formulate continual queries, that is, queries related to possibly continuous changes of data over time. The key idea behind PeerCQ is to let an arbitrary peer monitor the data and machines involved in a single query. Because each query has a unique identifier, this scheme fits nicely with the identifier-based routing protocols inherent to structured peer-to-peer systems. A drawback of this approach is that there may be many peers monitoring the same data or machines, in turn introducing a potential scalability bottleneck.

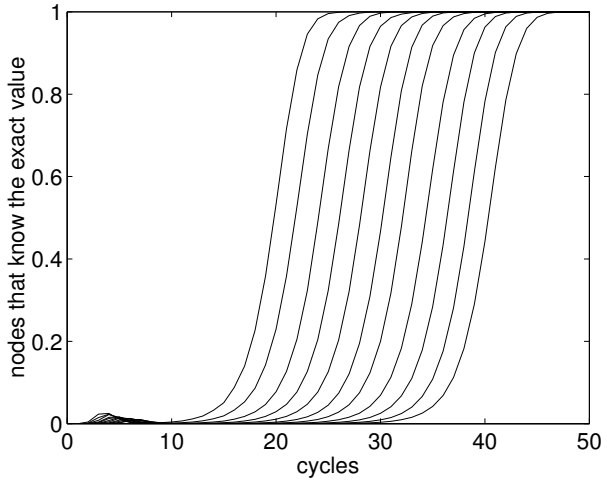
Scalable liveness detection has very recently been proposed in [17]. The essence of the proposed protocol is to offload the number of probes for a single device to its probing processes. The latter are dynamically organized into an overlay network by letting the device, on each probe, return the addresses of the last N processes that probed it as well. This approach will permit a probing process to reduce its probing frequency, because it will be informed by one its peers whenever the status of the monitored device changes.

6.2 Determining the size of a network

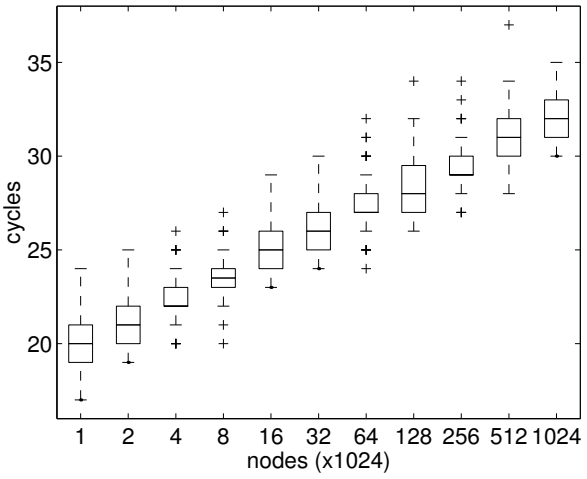
In Section 5.5 we briefly mentioned the possibility of finding the number of nodes in a network by calculating the average of a *peak distribution* $\mathbf{x} = (1, 0, \ldots, 0)$, that is, the distribution where one node holds the value 1 and all the others hold 0, and using $1/\overline{x}$ as an estimate of the network size N. Now we will take a closer look at this method.

The averaging algorithm we discussed in Section 5 has two nice properties: (1) the average of all node values does not change with the number of cycles, and (2) the variance is dropping at rate ρ^k, where k denotes the number of cycles, and ρ is slightly smaller than 0.4 (see Figure 14(a)). In other words, when the averaging algorithm is applied to the peak distribution, node values converge exponentially fast to $1/N$. To get an idea about the actual performance of this approach we have run a number of experiments using the idealized averaging algorithm. We have focused on two aspects: accuracy and speed.

First, we looked at how the number of cycles affects the number of nodes that know the *exact* size of the network. The averaging algorithm was applied to networks of size varying between 2^{10} and 2^{20} nodes. After every cycle all nodes whose values (after rounding) were equal to the size of the network were counted. Each experiment was repeated 100 times and results were averaged. They are shown in Figure 16(a). We can see that for networks with size ranging from 2^{10} to 2^{20} nodes there are about 25-45 cycles needed for full convergence to the exact value N. (More precisely, as we represent real numbers by standard 64-bit floats, the "exact value" means here the result of rounding to the closest integer.)



(a) The proportion of nodes that know the exact network size as a function of the number of cycles k and network size N. Consecutive lines (from left to right) correspond to networks with $2^{10}, 2^{11}, \ldots, 2^{20}$ nodes.



(b) The number of cycles needed for all agents to learn the approximated network size. Boxes have lines at the lower quartile, median, and upper quartile values. The whiskers (lines extending from each end of the box) show the extent of the rest of the data. Outliers are data with values beyond the ends of the whiskers.

Figure 16: Network size estimation

Although it is quite impressive that one can find the exact number of nodes with help of a simple averaging algorithm, in practice much smaller accuracy is sufficient (e.g., 1%). To get a better insight into the relation between this limited accuracy and the number of required cycles we run simulations until *all* the nodes knew the approximate network size. Again, for each setting 100 runs were performed. The results are summarized in Figure 16(b). We can see that the average number of required iterations dropped from 25-45 to 20-32.

Notice that due to its fast convergence rate, the averaging algorithm can be regularly restarted on a fresh peak distribution (e.g., every 100 cycles). In this way all the nodes are constantly aware of the network size.

6.3 Monitoring total amount of resources

A simple equation: $\sum x = N\overline{x}$ now has an important implication: once we know how to find N and the average $\overline{x}$, we immediately know how to find the sum of x's. And in the light of results presented in the previous sections we already know that both ingredients (i.e. N and $\overline{x}$) can be found in a small number of cycles and with an arbitrary accuracy. Therefore, we have an efficient algorithm that can be used for finding the sum of all values that are stored by nodes.

In the context of network monitoring it means that one can constantly measure the total load of all the nodes, the total capacity of their disk drives, the total number of files or amount of data stored, etc.

6.4 Monitoring migration of agents

Yet another application of aggregation algorithms for network maintenance is measuring the number of agents that joined the network within the last *epoch*. By an epoch we mean here a fixed number of cycles (e.g., 1000). Counting agents that joined the network is simple: every agent has to keep one bit of information: 1, if it joined the network within the current epoch; 0, otherwise. At the end of every epoch the sum of these bits is found, and 1's are set to 0's.

Combining this information with the network size one can calculate the number of agents that left the network during the epoch. This way the fluctuation of the network can be monitored.

An additional mechanism could be built-in into every agent: whenever the number of failures that are observed by a single agent exceeds a certain threshold, this agent may rise an alarm.

6.5 Alarm processing

As a next example, consider the problem of efficiently broadcasting an alarm signal. As we mentioned, broadcasting is just a special case of computing and disseminating the maximum value across the network. In particular, we let $\mathbf{a}_0 = (a, 0, \ldots 0)$ where $a \geq 0$ denotes the alarm value generated by node 1.

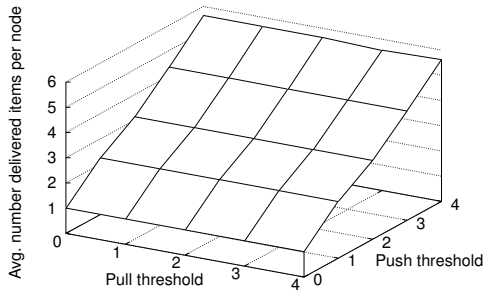
To ge beyond simple broadcasting, in this example we extend the topology manager protocol as follows. The descriptors that are stored in the partial views of nodes will contain not only the address and hop count, but also information if the given node have seen the alarm sig-

nal at the time of creation of the descriptor. In other words, the node descriptors contain the approximation of the aggregate of the node.

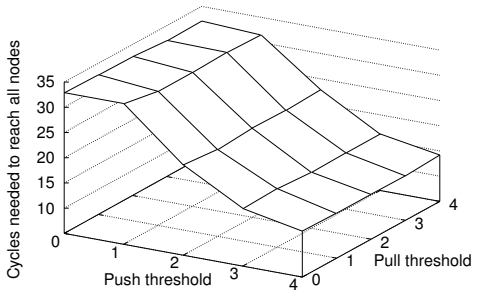
It has been shown already that broadcasting can be done super-exponentially fast [34]. However, this dissemination speed comes at a price: there are many nodes that receive the alarm value more than once. We can improve this situation by taking the hop count of a descriptor into account.

Instead of randomly selecting a neighbor from the partial view provided by the topology manager protocol, a node adopts the following selection policy. If a node has already seen the alarm signal, it will give preference to a peer whose descriptor (1) has value 0, and (2) is young. The reasoning is that most likely this peer has not yet seen the alarm signal, whereas other peers with a zero-valued, but old descriptor are more likely to have already been contacted.

For the same reason, if a node is yet unaware of an alarm, it will give preference to a peer whose descriptor is old (has a large hop count). Note that the partial view of nodes who are unaware of an alarm contains only zero-valued descriptors.



(a) The average number of delivered alarms per node.



(b) The dissemination speed of an alarm signal.

Figure 17: Performance of alarm signal propagation.

The effect of this *selective push-pull policy* can be observed in Figure 17(a). The z-axis shows the average number of delivered alarm signals per node. The x-axis shows the *pull threshold*, which is relevant only for a node that is unaware of the alarm. The pull threshold is the minimal hop count that a descriptor should have before its associated peer will be contacted. Note that the pull threshold does not affect the average number of delivered alarm signals: no node will ever see the value more than once. Likewise, the y-axis shows the *push threshold*, which is relevant only for nodes that already have seen the alarm. The push threshold is the maximum age a news item can have in order to for its associated peer to be contacted. From this figure, it can be seen that the push threshold is indeed the only factor for determining the average number of delivered alarm signals per node.

Lowering the number of deliveries comes at a price: the number of cycles needed to disseminate the alarm to all nodes increases. Again, it is the push threshold that is the dominant factor, as shown in Figure 17(b). In this case, the z-axis shows the number of cycles needed to deliver the alarm to all nodes. The x-axis shows the push threshold; the y-axis the pull threshold.

7 Use of ant-based methods for performance monitoring in dynamic networks

Precise and up-to date measurements are essential in order to do long-term network planning and management, and short-term traffic control and engineering. In today networks, measurements are typically initiated and controlled from a central position/role in the network. For instance, observation and collection of statistics are done in a distributed way, data aggregation is also done locally. However, the resulting data are sent to a central management entity (center) that extracts the global information about the network state and manually activates the proper countermeasures. In order to do the management as correct and efficiently as possible, the statistics must be as precise and up-to date as possible. To avoid excessive amounts of data, the aggregation must be effective, without throwing away any valuable information.

The purpose of measurements consists in monitoring the behavior and the quality of delivered services. In this section we investigate weather a CAS that is designed to provide a specific basic network service can also provide additional metrics and measured that can be used in turn to realize the *online and adaptive monitoring* of the service. The specific case considered in this section is the routing function under the assumption of using algorithms inspired by foraging ant behaviors, that is, the so-called, *ant-based* routing algorithms (see also Section 1 and Appendix A.2).

The considered ant-based routing models use observations of network conditions directly in their objective function (cost value), and indirectly in their use of pheromones. If the ant-based method is used for adaptively controlling the routing, observing the model indices are of less importance. However, if the network operator is reluctant to let ant agents control their network configuration, the indices of ant-based routing model can be used to obtain information about the network condition as important input to traffic engineering in combination with other measurement data.

The scope of this study is initially fixed packet switched network like an IP-network. The network will experience changes in topology caused by link and node failures, mis-configurations, and redesign. The data traffic served by this network is very bursty and will influence the routing function when the routing metric is delay and bandwidth sensitive. The reason for starting with a less dynamic network than ad-hoc network is that the ant-based routing algorithm is added just to provide more accurate information network state to the network operator. No central network operator of an ant-net is foreseen. However, if the ant-based routing algorithm contains information that could improve the global behavior of the network, this should result in rules and algorithms that are instantiated on the nodes even in an ad-hoc network. This means for instance that if a reactive-proactive routing approach is taken, like those described in Section 1, the indices of the routing algorithm are observed, and changes in these indices could trigger an alarm and activate some local rule, or spread this alarm in the neighborhood by diffusion or using an approach like the one described in Section 6.

This section starts with a high level overview of monitoring approaches used today. This is followed by a section that takes a closer look at different observables, the *CAS indices*, observable in ant-based routing algorithms. The transient behavior of the ant-based routing models was studied in highly dynamic networks with frequent changes in both topology and traffic conditions. Two ant-based approaches, *AntNet* [37, 38] and *Cross-Entropy assisted Ants*

(CEAnts) [79, 159, 160], were chosen because they have both previously reported good results. The preliminary results are promising with respect to their abilities to adapt to network changes and to provide "network health indices". The specific focus is on identifying observables that might be applicable for monitoring the network condition with respect to the routing function.

7.1 Performance monitoring

In general, monitoring is supervision of: resource usage, utilization and availability. Examples of resources in the BISON context are information content (documents, video, music, etc.), processing power, link capacities, routers and terminals, battery power. Monitoring is a general term also applied in many other fields, e.g., a general definition can be found in the field of environmental protection [157], where monitoring is defined as *the periodic oversight of a process, or the implementation of an activity, which seeks to establish the extent to which input deliveries, work schedules, other required actions and targeted outputs are proceeding according to plan, so that timely action can be taken to correct the deficiencies detected*. The equivalent of *processes* in BISON context are aspects like: traffic profiles, service usage, server, scheduling, load sharing. The *timely actions* corresponds to network management, and *deficiencies* are traffic overload, route flaps, link/router/server down, etc.

More specifically, network and service monitoring is the oversight or supervision of the resources relevant for BISON. The *performance* monitoring function is relevant and important both from a user and provider perspective. The user (consumer) is interested in monitoring the quality of the delivered service, while the provider is concerned with maximizing the resource utilization when delivering service within the given quality guarantees. The service quality constraints are part of a Service Level Agreement (SLA) between consumer and provider. In the SLA, the monitoring function should also be described in order for the consumer and provider to have common understanding of what the service quality is and how this should be verified through measurements.

Monitoring objectives in BISON is considered to be:

- Service dependability (availability and reliability) in ad-hoc and overlay networks
- Performance and utilization of network elements and service platforms

BISON considers monitoring in both user and provider perspective. It is important to emphasis that performance monitoring is NOT network monitoring in the sense of surveillance of the content of communications and network usage of individuals.

Monitoring in packet-switched networks

There are mainly two approaches for performance monitoring of networks and service platforms: *Active monitoring* where test packets are submitted and their response (status, delay, route, etc.) are observed, and *Passive monitoring* where server logs and packet counters are sampled, or packets are (partially) captured.[12]

[12] A lot of work is being done in the field of monitoring. For the BISON project the particular interest is the monitoring and measurement techniques for IP based (i.e. packet) network, see for example [19] for state-of-art and [69] for an excellent tutorial on IP monitoring.

1. **Active monitoring**: Active means sending test traffic/packets and observing the performance of this traffic/packets. In IP networks active techniques are in widespread use both on application, transport and network layers. Examples are use of built-in functions on end-user equipment (ICMP ping [123]) trace-route [148]) and dedicated routers (RIPE NCC [126]) or route functions (Service Assurance Agents [26] of the Cisco IOS).

2. **Passive monitoring**: Passive means observing user traffic by counters or traffic/packet capturing. Sampling of interface counters on routers (e.g. by SNMP), reading server logs, packet capturing by sniffers are examples of passive techniques. Major challenges include reducing and accumulating data without throwing away valuable information input to traffic engineering, accounting, security surveillance, etc. A huge number of both commercial and free products are available.

There has also been done work on combined active and passive techniques, e.g. ATM Forum has defined OAM cells to be sent periodically (actively) dependent on the traffic load (observed passively). A similar approach is proposed by Lindh [103] for IP networks but not yet adopted by the IETF.

Performance monitoring and measurement techniques are part of the work in the standardization bodies. Both Internet Engineering Task Force, IETF (www.ietf.org) and ITU-T (www.itu.org) address these issues. See Appendix B for more details on this.

Monitoring network health by looking at CAS indices

The reason for looking at CASs for monitoring is that they will (potentially) provide essential global information stored locally, and make it easier for operational personnel to accept "letting CAS into the network". It is expected that it is easier to accept CASs for monitoring than for taking over the control.

In this section ant-based routing algorithms in dynamic networks are considered as an example of a function to be solved by a CAS. The ant routing table with its *pheromones* can be considered as snapshot or sample of the current network state. This network state reflects both traffic load and traffic matrix, network capacities, and availability of the network elements. This means that the network operator will receive importance information for traffic engineering by monitoring the indices of the CAS that proposes the routing.

Using this approach for monitoring might be considered as a combination of active and passive techniques where the ants are active test packets exploring the network and looking for the best routes (best in accordance to some quality attributes like the number of hops, total delay, available bandwidth, etc.) and where the ant routing tables and pheromones are passively sampled by some control center.

In Figure 18 an illustration of how CAS designed for routing might serve as input to monitoring function. The pheromones in the ant routing table is updated, The control center is polling the routing tables and pheromone values and receives an alarm then a rapid increase in the grade of converge is observed. Furthermore, the control center sends out ants to search for routes and the "convergence index" is observed in the returning ants. This index indicates the grade of convergence, i.e. the strengths of the signal or goodness of the path.

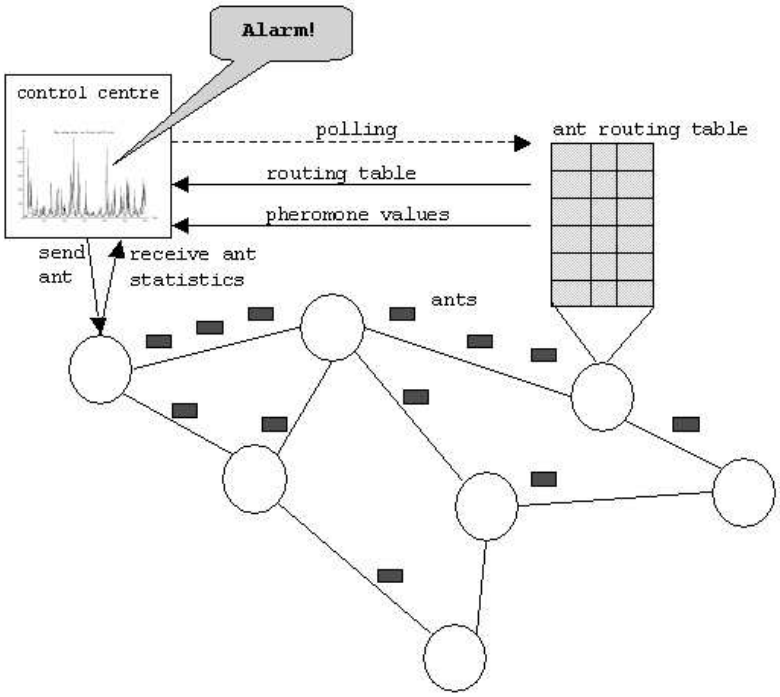


Figure 18: CAS for monitoring

7.2 Ant-based routing

Before ant-based routing models [134, 37, 79], can be applied for control of routing in real networks, it is important to understand more about their behavior. For the BISON project this mean to understand their transient behavior in highly dynamic networks, i.e. networks with frequent changes in both topology and traffic conditions. The knowledge about the transient behavior includes knowledge about the model's ability to solve the optimization task, i.e. find the best route, measured as the goodness of the solution and the rate and grade of convergence. It is also important to know that the routing is stable under transient (noisy) network conditions, but at the same time that new routes are found when the changes are of more permanent nature (permanent according to the time granularity condition valid for a specific network dynamics). Should ant-based routing methods substitute traditional routing approaches like OSPF, ISIS, BGP, it is essential to trust that the routing is (close to) optimal, is stable, and robust to changes in conditions.

As also discussed in Section 1 and in Appendix A.2, the concept of using multiple agents with a behavior inspired by foraging ants to solve problems in telecommunication networks was introduced by Schoonderwoerd & *al.* in the domain of circuit-switched networks [134] and by Di Caro and Dorigo's *AntNet* [37] in the domain of IP networks. Both these algorithms are implementations of Ant Colony Optimization (ACO) algorithms [46, 44]. While Helvik and Wittner [79, 159, 160] introduced *CEAnts*, an approach using ant-agents for finding primary and backup paths and which has been designed on the cross-entropy method [130].

In this section the focus is on routing (only primary paths) in datagram networks. Two previously proposed ant-based routing approaches, AntNet and CEAnts are studied in more detail. They both make use of a set of concurrent distributed agents to solve the routing problem where the agents build routing tables from information obtain while exploring the network. In this section the AntNet and CEAnts are used for routing in highly dynamic networks.

The algorithm

The routing problem is to find the best path from node s to d in a network, represented by $G(t)$, a unidirectional connected graph at time t. The $V(G(t)) = \{v_i\}$ are nodes (vertexes) in $G(t)$, where v_i is node i. The $E(G(t)) = \{e_{ij}\}$ are the links (edges) in $G(t)$, where e_{ij} is the link between node i and j, $v_i - v_j$. N_e is the number of links in each node. The $c(e)$ is the cost of link e, and $L(\pi) = \sum_{\forall e \in \pi} c(e)$ is the total cost of path π. More specifically, the path $\pi_{t,s}^{(d)}$ is the path (trajectory) found from source node s to destination node d after iteration t.

The routing problem from source s to destination d is either

- *hop-by-hop routing* - the routing information in the intermediate nodes from s to d is destination specific and contains information about the best route from the current intermediate router to the destination
- *virtual connection* - the routing information in the intermediate nodes is source and destination specific and contains the information about the best route all the way from s to d.

In the following, the virtual connection routing problem is used as an example. However, the same principle applies for the hop-by-hop routing as well. The cost of the different paths from s to d changes over time due to topology changes (node and links move, appear, and disappear) and changes in traffic pattern.

The ant-based routing system is an adaptive algorithm with a random mechanism, i.e. an auxiliary stochastic process like a Markov chain. The random process gradually change the routing probabilities.

The algorithm consists basically of two main steps:

1. Generate random paths $\pi_{t,s}^{(d)}$ from source to destination on the graph $G(t)$ using the random process $\mathbf{p}_t$ and calculating the object function $L(\pi)$ (the cost values).
2. Update $\mathbf{p}_t$ on the basis of the data collected along $\pi_{t,s}^{(d)}$.[13]

$$f : \mathbf{p}_t \leftarrow \mathbf{p}_{t-dt} \tag{17}$$

where dt is the increment, and f the change of measure. This is different for the two approaches

- *AntNet* - the f is based on reinforcement learning [142] where the learning factor r includes information of the last, and historical, cost values.

[13] To simplify notation the ant species index is neglected in the following.

- *CEAnts* - the f is the change of measure governed by the optimal Cross Entropy method [131]. The CEAnts uses a distributed version of the Cross Entropy method where the f is approximately optimal.

More specifically, this means that for each iteration t the agents do:

- *Forward search* - at each node i along the path from the source node, s, to the destination node d choose the next edge e_{ij} at random according to the routing probabilities in node $i, p_{t,ij}^{(d)}, \forall j \in v_i$,
- *Path evaluation* - determine the cost value, $L(\pi_t)$ of the path of iteration t, π_t
- *Backward updates* - return to source node s and update the routing probabilities in each node along the path λ_t found in the step 1.

The AntNet and CEAnts basically follows the same steps. However, a few differences exist.[14]

Forward search

The forward search uses the following routing probability at time t in node d

$$p_{t,ij}^{(d)} = (T_{t,ij}^{(d)} + x_{t,ij}^{(d)}) / \sum_{\forall k} (T_{t,ik}^{(d)} + x_{t,ik}^{(d)})$$

where $T_{t,ij}^{(d)}$ and $x_{t,ij}^{(d)}$ are the pheromone value and local traffic statistics, respectively, at time t over interface j in node i for destination d.

AntNet uses local traffic statistics at edge e_{ij} in node i to adjust the routing probabilities of node i, v_i by the $x_{t,ij}^{(d)}$. The traffic statistics are the queue lengths. This introduces a short feedback loop that will decrease the probability of selecting the path with the best pheromone value when this path is highly loaded, i.e. the queue is non-zero. The CEAnts use $x_{t,ij}^{(d)} = 0$ for all t, i, j, d. This implies a longer feedback loop and potentially a less "nervous" system.

Path evaluation

At the destination node the cost value $L(\pi_t)$ of t'th path found, π_t, is calculated. Based on this cost value, a performance function h_t is obtained that includes the cost value and some cost value history. AntNet and CEAnts use different strategies for this:

AntNet - remembers the historical cost values through an estimate of the average, μ, standard deviation, σ, and best value W_{best} over an observation window $\mathcal{W}$. The size of $\mathcal{W}$ is given

[14] The description of the ant-based routing includes specific details of the AntNet based on [37], and the CEAnts algorithm based on [79, 159, 160].

by the memory factor η, where $|\mathcal{W}| = 5(c/\eta)$. The best value in $\mathcal{W}$ requires that the entire $\mathcal{W}$ must be stored. The performance function is (see detail in [37]):

$$h_t = c_1 \frac{W_{best}}{L(\pi_t)} + c_2 \frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) - (L(\pi_t) - I_{inf})} \tag{18}$$

where

- W_{best} - best cost value in moving exponential window $\mathcal{W}$
- $I_{sup} = \mu + z(\sigma/\sqrt{|\mathcal{W}|})$
- $z = \frac{1}{\sqrt{1-\gamma}}$
- $\mu_t = \mu_{t-dt} + \eta(L(\pi_t) - \mu_{t-dt})$
- $\sigma_t^2 = \sigma_{t-dt}^2 + \eta(L(\pi_t) - \mu_{t-dt})^2 - \sigma_{t-dt}^2)$
- $I_{inf} = W_{best}$
- $c_1 = 0.3$
- $c_2 = 1 - c_1 = 0.7$

CEAnts - remember the historical cost values through an autoregressive formulation with a memory parameter β. This formulation enables a compact representation of previous cost values and they are weighted decreasingly as time goes by. The performance function is (see details in [79]):

$$h_t = \beta h_{t-dt} + (1 - \beta) e^{-L(\pi_t)/\gamma_t} \tag{19}$$

The γ_t is the scaling parameter that is the result of the optimization of the change of measure f in the routing probability matrix. In order to avoid storage of all (or a part) of previous cost values, the γ_i is calculated through a first order Taylor expansion:

$$\begin{aligned}
\gamma_t &= \frac{B e^{\frac{L(\pi_t)}{\gamma_{t-dt}}} + L(\pi_t)}{1 + \frac{L(\pi_t)}{\gamma_{t-dt}} + e^{\frac{L(\pi_t)}{\gamma_{t-dt}}} (A - \rho \frac{1-\beta^{M+1}}{1-\beta})} \\
A &\leftarrow \beta A + (1 + \frac{L(\pi_t)}{\gamma_t}) e^{-\frac{L(\pi_t)}{\gamma_t}} \\
B &\leftarrow \beta B + L(\pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} \\
\gamma_t &\leftarrow \gamma_{t-dt} \\
M &\leftarrow M + 1
\end{aligned}$$

The node only needs to store γ, A, B, and M instead of the complete observation window.

After iteration t, the agents are carrying information about the path found, π_t. In addition, information about the observed cost value and historical values:

AntNet - the r_t factor calculated from $r_t = s(h_t)/s(1)$ where $s(x) = 1/(1 + e^{a/(xN_k)})$, where $N_k \leq N_e$ is the number of links in node k. An upper limit of r_t, e.g. $r_{max} = 0.85$, is recommended to avoid too strong convergence to one path.

CEAnts - the temperature γ_t and cost $L(\pi_t)$

Backward updates

The backward agents updates the pheromones T_j (the destination d, the iteration/time t and node i indices are suppressed) and the corresponding $p_j = T_j / \sum_{\forall k} T_k$. AntNet and CEAnts use different strategies for updating the pheromones:

AntNet - the pheromone is given as

$$T_j = I(\{j\} \in \pi_t) r_t + (T_j - r_t T_j) \tag{20}$$

This means that each node must store a set of T_j, one for each destination d and for each outgoing link j

CEAnts - the pheromone is given as

$$T_j = I(\{j\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} + A_j + \begin{cases} -\frac{B_j}{\gamma_t} + \frac{C_j}{\gamma_t^2} & \frac{1}{\gamma_t} < \frac{B_j}{2C_j} \\ \frac{-B_j^2}{4C_j} & \text{otherwise} \end{cases} \tag{21}$$

where

$$\begin{aligned} A_j &\leftarrow \beta A_j + I(\{j\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} (1 + \frac{L(\pi_t)}{\gamma_t}(1 + \frac{L(\pi_t)}{2\gamma_t})) \\ B_j &\leftarrow \beta B_j + I(\{j\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} (L(\pi_t) + \frac{L(\pi_t)^2}{2}) \\ C_j &\leftarrow \beta C_j + I(\{j\} \in \pi_t) e^{-\frac{L(\pi_t)}{\gamma_t}} \frac{L(\pi_t)^2}{2} \end{aligned}$$

This means that each node must store a set of A_j, B_j, C_j, one for each destination d and for each outgoing link j

The objective function

The objective function considered in this study is the end-to-end delay of the path $\pi_s^{(d)}$ from source node s to destination d.

$$L(\pi) = \sum_{\forall e \in \pi} c(e) \tag{22}$$

where $c(e)$ is the cost of link e. In this section $c(e)$ is the delay experienced by the agent traveling over this link, including queuing and processing at the originating end.

7.3 The simulation model

To study the two algorithms, AntNet and CEAnts, a simulation model of a dynamic network with topology changes and changes in traffic patterns is developed. The focus is on Ad-hoc network with datagram transport (i.e. IP networks) with irregular and dynamic topology. The model also applies to studies of in overlaid networks.

The network model is developed for studies of transient effects of ant-based routing in highly dynamic networks. In order to focus on how the routing adapts to the network dynamics it is important to balance the level of details. Too many details might confuse effects of physical constraints and data traffic protocol behavior and constraints with effect of ant-based routing behavior. On the other hand the model has to include sufficient level of details to capture and describe the main contributions to the network dynamics.

The details of the simulation model are reported in this section. The model includes a reference system where users and terminals can operate, move, appear and disappear. The reference system is a cell structure. A terminal can forward agents that updates the routing information on backtracking after the target destination is reached. Both source, destination and forwarding terminals are denoted *nodes*. This cells and nodes include features that enable changes in the network topology and in the data traffic patterns.

The underlying cell structure

To enable the modeling of user and terminal mobility, i.e. node mobility, it is necessary to define a reference (coordinate) system. Here a cell structure $\vec{\chi} = \{\chi_1, \cdots\}$ is defined where each cell χ_i contains

- position - relative to other cells, (x, y)
- edges (interfaces) to neighbor cells, χ_i.
- physical constraints to neighbor cells
- (current) number of visiting nodes (0 or more)

The model does not have hierarchical cell structure. The number of edges per cell is (in current implementation) a global variable, N_e and will be the same for all nodes. However, it is still possible to model a system with different number of edges by letting N_e be the maximum number of edges and reduce the number for cells with less than N_e by introducing physical constraints.

The nodes

The nodes model the user terminals (e.g. in ad-hoc networks) or peering points in an overlaid network (e.g. peer-to-peer). Each node binds to one cell at the time. A node has the following features:

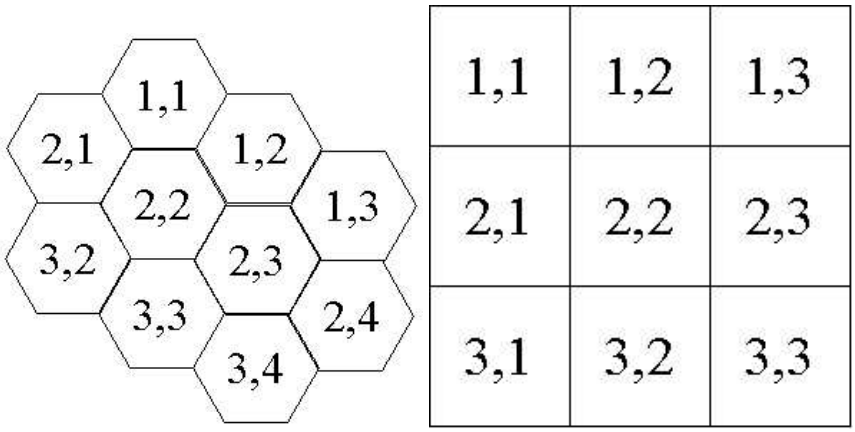


Figure 19: Cell structure with (a) 4 edges, (b) 6 edges

- forwarding[15] search ants according to the routing probabilities - either uniform if exploration ant or by normalized pheromone values if not.
- forwarding backtracking ants who updates the pheromone values
- if node is the destination of an ant, update score value and CAS indices.
- if node is the destination of an backtracking ant (the source of the ant), send the ant to receiving process of ant nest, (defined in the following page).

Each node stores a lot of information to support both ant-based routing in general, and AntNet and CEAnts, specifically. Node i contains:

- pheromone values, $T_{ij}^{(d)}$ over interface j in node i for destination d (these are local CAS indices for AntNet and CEAnts)
- routing probabilities, which is the normalized pheromone values, $p_{ij}^{(d)} = \mu_{ij}^{(d)} / \sum_{l=1}^{N_e} \mu_{il}^{(d)}$
- intermediate values for CEAnts (A, B and$C$)
- in destination node:
  - *AntNet:* score value statistics and history window
  - *CEAnts:* accumulated score value statistics

There is no hierarchical node structure, i.e. all nodes forward agents, store pheromone values, and can be both source and destination nodes. They may have dynamic behavior.

[15]The packets are forwarded to a cell and not a node because a node may move and disappear

The ants agents

The forward ant behavior is to search for a destination node and to return back to the nest along the reversed route updating the pheromone values.

The ants that are forwarded by the nodes are data packets carrying information necessary for the AntNet and CEAnts algorithms. The data packets may also model ordinary data traffic. The ant's logic is inside the nodes and not in the packets and hence these are not "agents". The following information is included in the packets:

- originating node for search (source address)
- terminating node for search (destination address)
- accumulated travel time to estimate end-to-end delay when destination address is reached or to discard packet if TTL is expired.
- accumulated route probability
- CAS indices for AntNet (r-factor) and CEAnts (temperature γ)

The ant nest

The ant nest is the node where the search for a destination is originated. This node have

ant generators - responsible for emitting ants that are searching for a destination node.

ant receivers - that receive and update statistic (e.g. score value and r-factor or γ-temperature) of the route found by the ant.

Structural dynamics

The networks structure or topology may change while the ants are searching for routes from source to destination. The cell structure is fixed but the physical constraints may change to model environmental changes. In addition to this, new nodes may appear, existing nodes may disappear or move to another cell, and old node may reappear. The scenario used as an example in this paper is illustrated in Figure 21, 22, and 23, and described in Section 7.4. This shows one example of topological changes that can be modeled.

Traffic load

In addition to structural dynamics, changes in the traffic load will have a significant influence on the performance of the ant-based routing algorithm. The most realistic approach is to generate data traffic by sending data packets similar to the ants. However this will impose a heavy burden on the simulator and will reduce the performance dramatically.

Since it is only necessary to model the influence of data traffic on the convergence and performance of the routing algorithm and not details in the data traffic itself it is possible to take an

alternative approach. This idea is to model each node as an $M/M/N_e$-system, and then randomly split this into $M/M/1$-system for each of the N_e interfaces. In Figure 20 this is illustrated. Traffic is offered to node i according to a Poisson process with intensity λ_i. These data packets are routed to one of the N_e outgoing interfaces according to the routing probabilities $p_{ij}^{(d)}$ where they receive a negative exponentially distributed service time with intensity $\mu_{ij} = 1/c_{ij}$ where c_{ij} is the link rate of interface j of node i.

Then, instead of modeling each data packet arrival epoch, only the ant arrival epochs are studied as an embedded Markov process. It is known from Burke's theorem [76] that in an $M/M/N_e$-system the departure process is Poisson. This means that the ant packets arrive each node along the route according to a Poisson process. Furthermore, it is known that the arrival, departure and a random observation process sees the same distribution of the number of customers in the system [97] and hence also the same waiting time distribution.

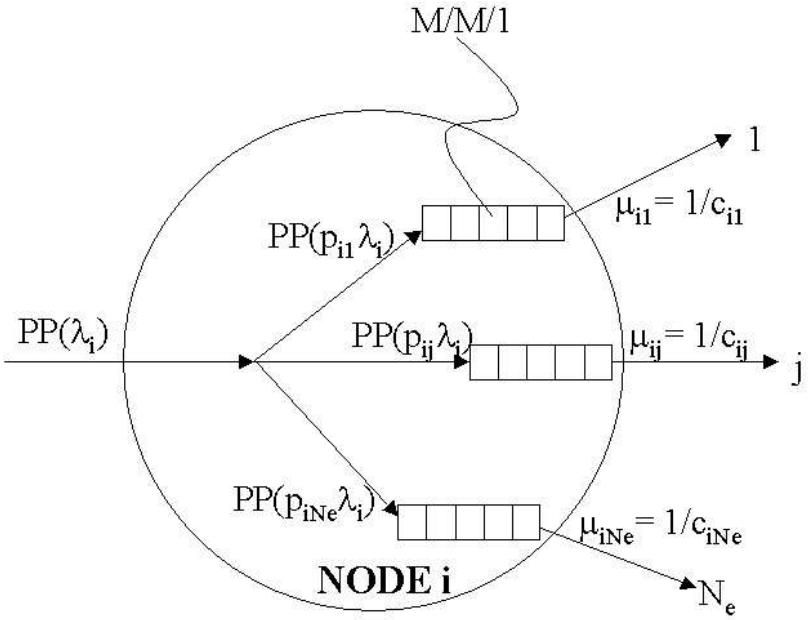


Figure 20: The traffic load at interface j in node i is modeled as a M/M/1-system

Under these assumptions the time through node i for an ant is the service time plus the random queuing delay sampled from the waiting time distribution for an $M/M/1$-system [97]. The delay distribution for interface j in node i is:

$$f_{ij}(t) = (\mu_{ij} - \lambda_i p_{ij}) e^{-(\mu_{ij} - \lambda_i p_{ij})t} \tag{23}$$

where $\mu_{ij} = 1/c_{ij}$ is the service intensity of interface i, λ_i is the arrival intensity to node i, and p_{ij} is the routing probability.

This means that when an ant arrive a node, it is routed according to the routing probabilities to interface i, and then a random delay D_{ij} is sampled from the distribution given in Eq.23.

$$D_{ij} = -\log U/(\mu_{ij} - \lambda_i p_{ij}) \tag{24}$$

where $U \in Unif(0,1)$ is a uniformly distributed random variate between 0 and 1.

The model's limiting assumptions are that the traffic load in each node is independent the other nodes and that the data traffic arrives according to a Poisson process.

7.4 Dynamic network scenarios

This section describes the scenarios used in the study of the transient behavior of AntNet and CEAnts with respect to their robustness and ability to provide monitoring indices.

Objective

The objective of this study is to observe the transient behavior of AntNet and CEAnts in a network with both high node/terminal mobility and few nodes (nodes move and are added/deleted) with high topology diameter, and low node/terminal mobility and many nodes (nodes are added/deleted) with low topology diameter.

To simplify the calculation of exact optimal solutions, the problem of finding best virtual connection from s to d in a grid network topology is studied. The optimal solution is easily found by use of the Dijkstra algorithm. Although this is a polynomial optimization problem, it is considered to be well suited for demonstration of the efficiency of AntNet and CEAnts, and for study of transient behavior and the use of score values and temperature for indication of cost value changes.

In order to study the ability to provide monitoring indices, the following are observed under the simulations as function of time t:

1. *Cost values index* - $L(\pi_t)$ (the trip time from s to d for path π_t,
2. *Convergence index* - r_t for AntNet and γ_t for CEAnts,
3. *Pheromones* - T_t (in normalized form equal to routing probabilities),
4. *Path probabilities* - $P(\pi_t)$.

The indices are influenced by changes in the topology and in traffic load variation.

Topology changes

The topology changes during the simulations. The topology alters between 3 phases: grid network (node connectivity 4), same network with major breakdown, and with partial restoration. The task is to find a optimal path from node (1,1) to node (8,8).

Phase I: Grid network The initial phase starts with a grid network where each interior node is connected to 4 neighbor nodes as illustrated in Figure 21. The links are bi-directional and may have different capacity. The values on the links are capacity (1 or 2 Mbit/s), hence the transmission delays are the reciprocal of these.

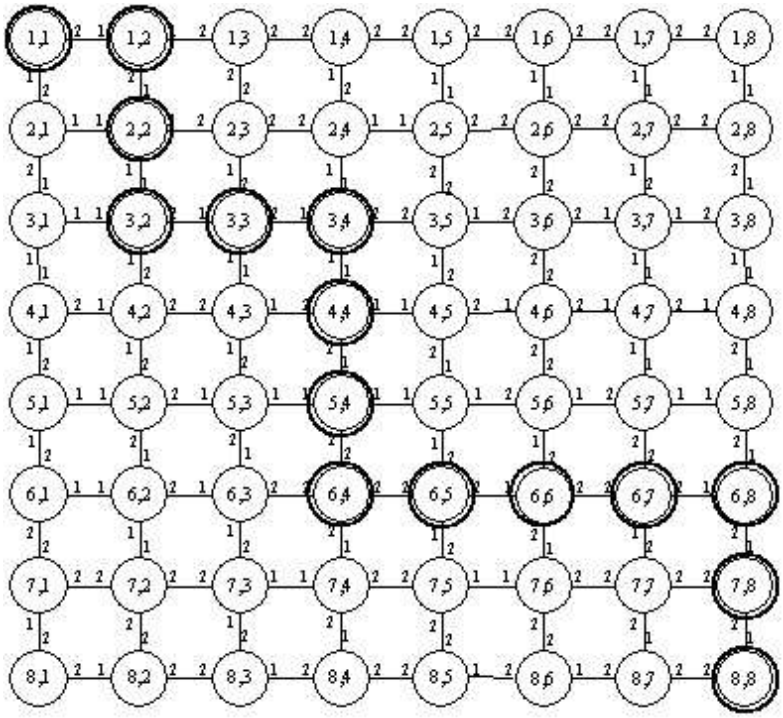


Figure 21: Phase I: Grid network

Phase II: Major breakdown A major breakdown is specified to observe the transient behavior of the routing method when the cost value is increased. The scenario is designed to cause a significant impairment by forcing the best path to almost double the length and the cost value. In Figure 22 this topology is illustrated and the best path is indicated.

Phase III: Partial restoration In the 3rd topology scenario a single node (1,5) is restored after the major breakdown. This partial restoration create a topology where a significant improvement in cost value of the best path exists. This new optimal path is not next to the optimal path in the major breakdown topology, see Figure 23. This challenges the routing algorithm to see if it finds this path even if the path in major breakdown topology has strongly converged.

Traffic dynamics

The traffic load are changed from 0 to 0.90 by changing the parameter λ_i in Eq. (23). In the preliminary studies the same parameter are used for all nodes, i.e. $\lambda_i = \lambda, \forall i$.

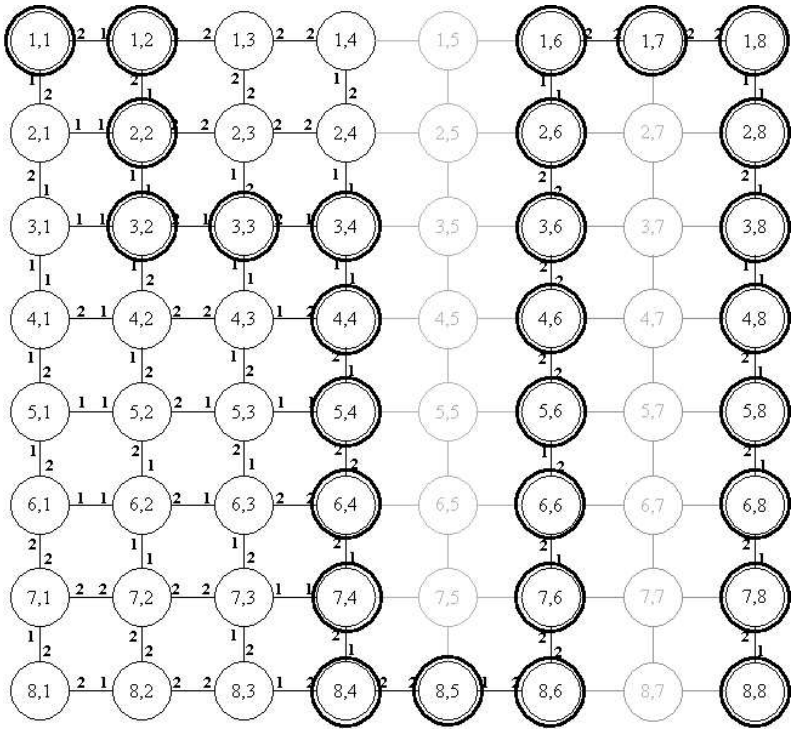


Figure 22: Phase II: Major breakdown

7.5 Metrics for performance monitoring: Preliminary results and observations

Study of transient properties of ant-based routing methods includes studies of transient effect in a network with frequent topology changes and variation in traffic load and matrix. Both AntNet and CEAnts are studied.

The network described in Section 7.4 has been simulated with changing topology between the 3 different phases, and changing the traffic load. The simulation experiments have included 10% exploration ants, i.e. agents that does not consider the current routing probabilities when selecting the next hop on their forward search but simply do random walk. They are included to be able to discover new and better paths even when they are off-track in steady state situation.

A sample of the results and some observations are included in the following.

Performance metrics

There are several CAS indices that are considered to be candidates for performance monitoring. In this section the transient behavior of pheromones, probability matrix, the convergence index (e.g. temperature in CEAnts) are studied as the network changes traffic pattern, traffic load, traffic matrix, topology (nodes add/del, move), users and information move. The objective is to investigate usability of CAS indices as monitoring information to feature management and operation of the network.

As a starting point, the following indices are studied:

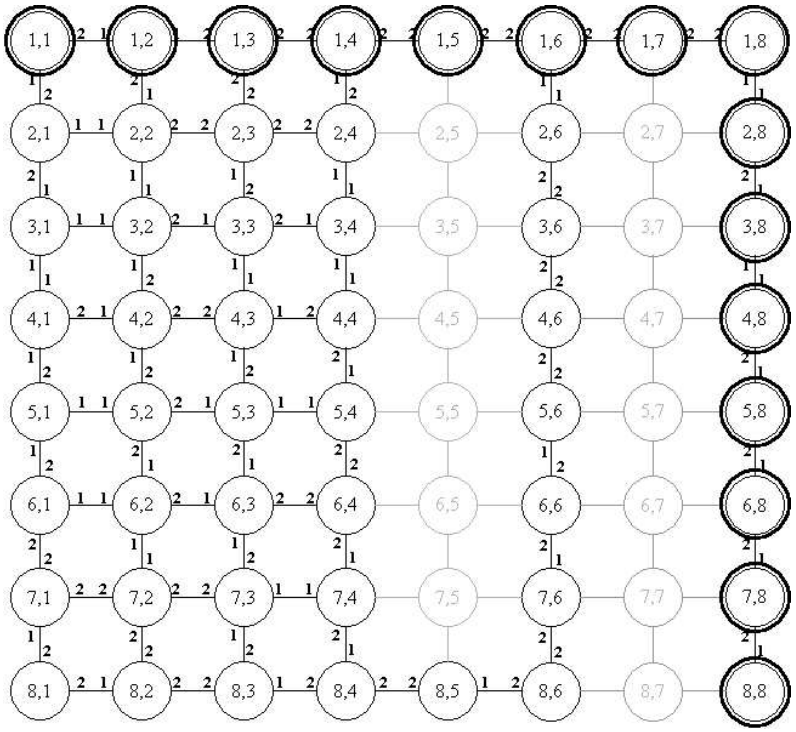


Figure 23: Phase III: Partial restoration

1. In the nodes:
   (a) Data routed by standard routing, ants by CAS: observe difference in routing tables (ideally both routing protocol should propose the currently best route.) no results are yet obtained from this.
   (b) Observe changes in the *pheromones*, T, and hence the routing probability matrix, $\mathbf{p}$. A sudden increase is indication of severe impairment, i.e. some node or link along the best path is removed, and sudden decrease can indicate that a new node or link is added and a better path has appeared.
2. In the agents:
   (a) Convergence index (r or γ, or path probability)
   (b) Cost value index
   (c) Path probability

Effect of major breakdown followed by partial restoration

Figure 24 shows results from simulation of AntNet and CEAnts with short memory factors in series of 10 replications each. The cost values, $L(\pi_t)$, and convergence factor (r_t for AntNet and γ_t for CEAnts), are plotted against the time t. Both the average values and the 95% confidence interval over the 10 replications are plotted. Figure 25 shows similar results using a higher memory factor.

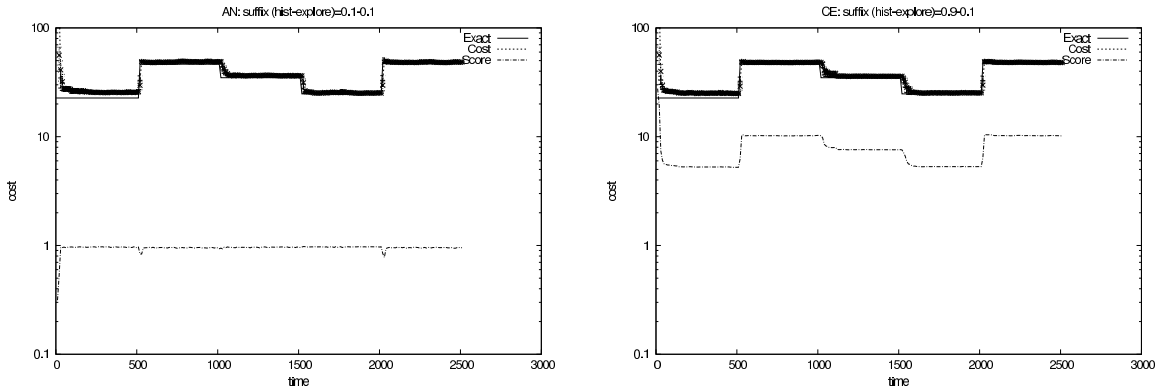


Figure 24: Short memory: The cost value and convergence indices as network topology changes with no data traffic load - plots of sample bins of 10 simulation replications

The simulations ran through phase 1, 2, and 3 as described above. The major breakdown was introduced at time 500 and partial restoration at 1000.

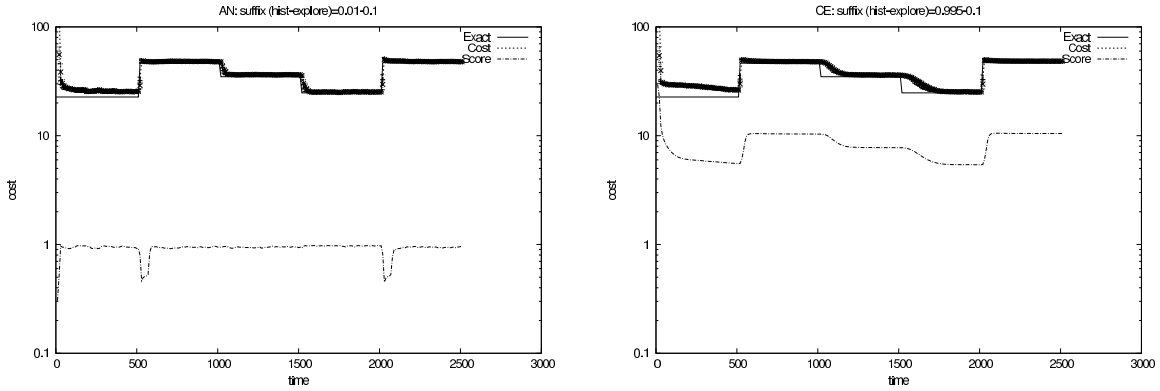


Figure 25: Long memory: The cost value and convergence indices as network topology changes with no data traffic load - plots of sample bins of 10 simulation replications

From these preliminary experiments it is observed that both AntNet and CEAnts react quickly to changes in the network as long as the memory is not too long. Some rough tuning of parameters have been done. It is also observed that after partial restoration the method finds the best path shortly after the topology restoration (where the cost is decreased again).

Variable traffic load

To study the influence of traffic on the routing algorithm, variable traffic load is added to the scenario in previous section. The simulation started with no traffic load and then stepwise increased the traffic up to 90% using the the Poisson model described in Eq. 23. At the end of the simulation (approx. at time 1100) the load is decreased to 0.5)

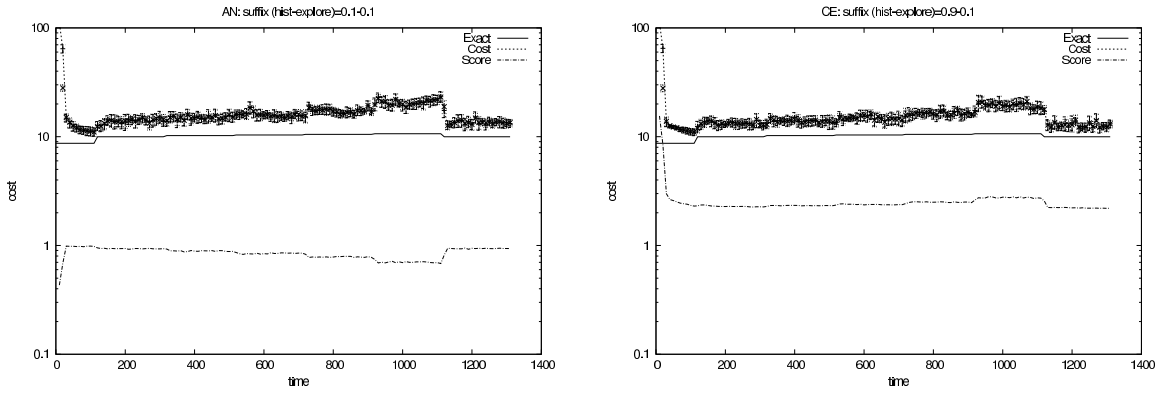


Figure 26: Short memory: The cost value and convergence indices as traffic load changes - plots of bins averages of 10 simulation replications

The results in Figures 26 and 27 show that the even though the cost values vary (because the object function is the end-to-end delay and hence strongly depends on the variation of traffic load), the routing algorithm is rather stable. For AntNet the memory factor does not change the behavior significantly, while CEAnts with long memory finds an improved solution at the end of the simulation period compared to both the AntNet and CEAnts with short memory.

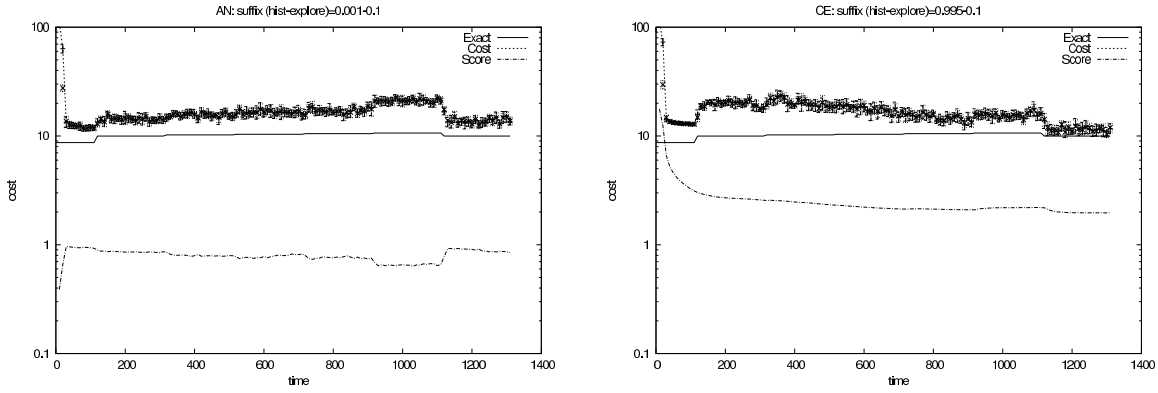


Figure 27: Long memory: The cost value and convergence indices as traffic load changes - plots of bins averages of 10 simulation replications

Frequent topology changes and variable traffic load

The results from a simulation series with frequent changes in topology and with variable traffic load is given in Figure 28 for short memory and Figure 29 for long memory. The topology changes are alternation between the 3 phases defined above. The traffic load changes over time by changing the λ_i in (23).

As was observed in Figure 24 and 25, both AntNet and CEAnts react quickly to changes in topology that both increases and decreases the object function (cost value). However, if the

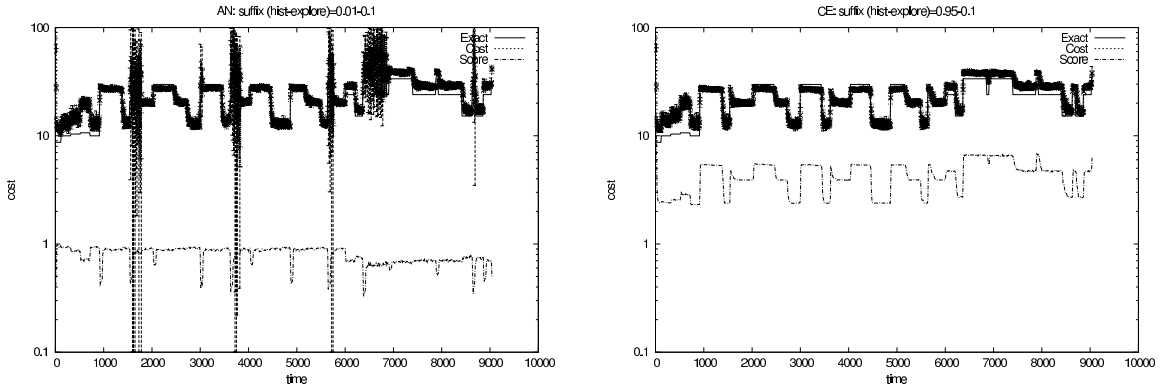


Figure 28: Short memory: The cost value and convergence indices in a highly dynamic network - plots of bins averages of 10 simulation replications

memory is too long some of the changes (the cost value improvements) will not be detected before the topology is changed again. It is an obvious challenge to balance the memory factor in order to rapidly detect all topology changes, and at the same time not end up with a solution far from the optimal, or a solution being to instable ("nervous") and be over-reactive to short-term changes in the traffic load. From the results it can be observed that the AntNet is rather unstable for some parts of the experiment where the traffic and topology changes rather rapidly[16]. Hence, fine tuning of parameters are necessary. The CEAnts seems to be more stable in these situations and less sensitive to fine tuning of parameters.

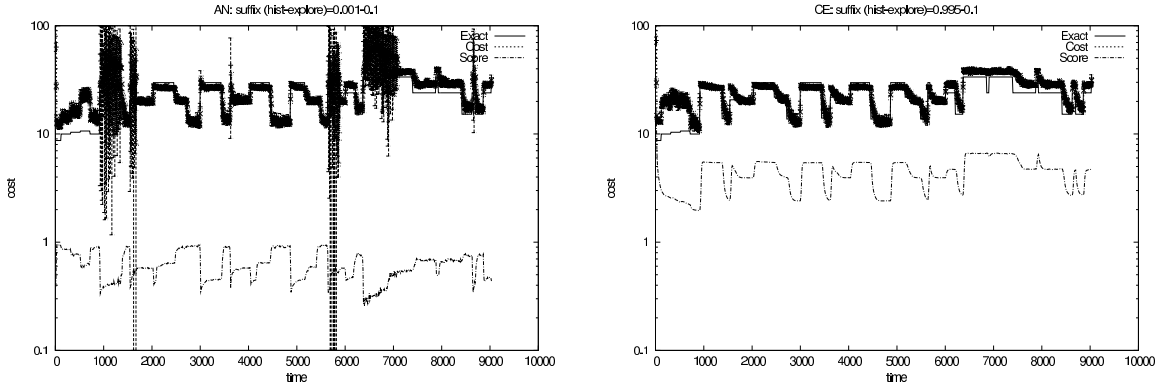


Figure 29: Long memory: The cost value and convergence indices in a highly dynamic network - plots of bins averages of 10 simulation replications

[16]The studies in this section have not used the local load statistics in AntNet

Path probability

The probability of a path (i.e. the product of the routing probabilities over the path) is

$$p(\pi_t) = \prod_{v=s}^{d} I(\{ij\} \in \pi_t^{(d)}) p_{t,ij}^{(d)}$$

In Figure 30 the path probability of each path found is plotted as a function of the simulation time for the case in Figure 29 above. The path probability is plotted for average of 10 simulation experiments together with the cost value and convergence index.

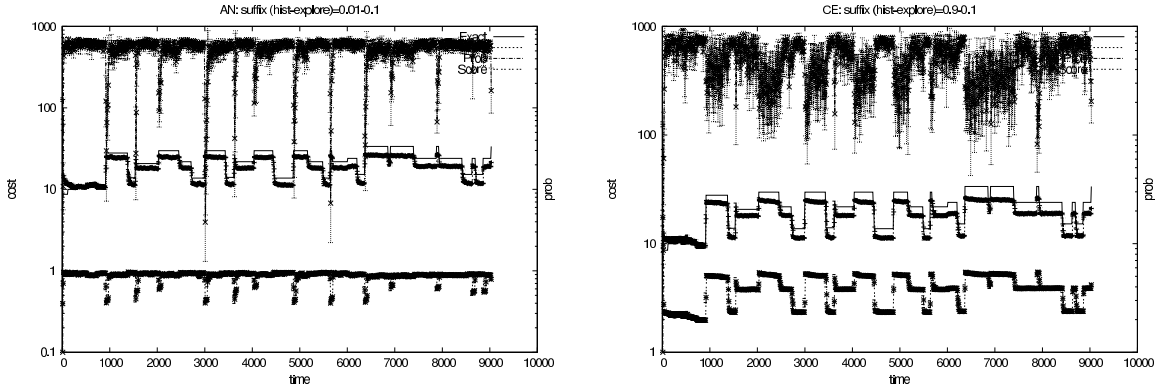


Figure 30: Path probability plotted with cost values and convergence index in highly dynamic network - average bin over 10 simulation replications

As expected, the path probability drops significantly when the network condition changes and new paths must be explored.

When a path is well established as a good path and alternative paths are not attractive, the path probability is close to 1. From Figure 30 it is observed that the path probability converge to 1 and stay close to 1 when little exploration of the search space is conducted. The more exploration, the more "noisy" the path probability appears, i.e. not all agents follow the same path.

In Figure 31 it is zoomed in on 2 topology changes. For AntNet (left) it is observed that the path probability drops when the cost value is decreased, but no significant effect is observed when the cost value is decreased (a better path has appeared). For CEAnts (right) the path probability drops significantly every time a new good path is found. This is valid both in the case when a better path is found under unchanged network conditions, and when a new path is found because the network conditions have changed.

The path probability will not change it the case of improved cost values when the new path is partially overlapping with previously found good paths.

In summary, the path probability could be used as an indication of exploration (instability) but it is not evident that it is suitable for detecting changes in the network conditions.

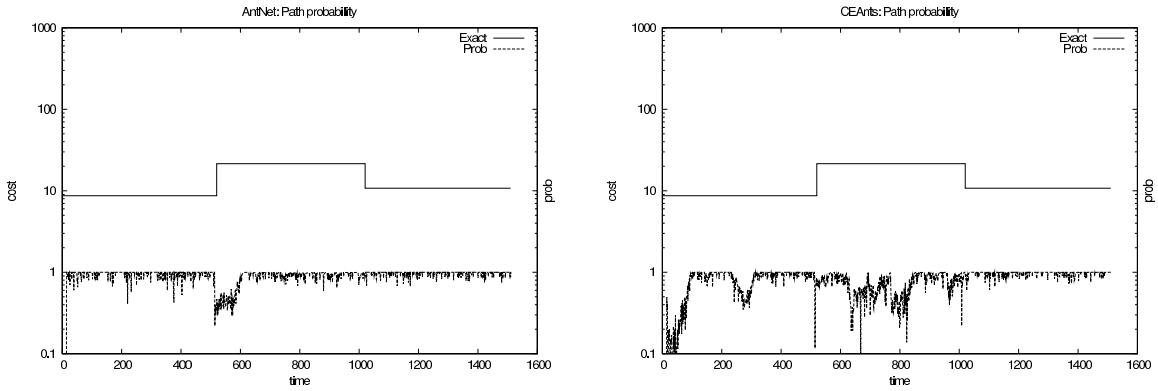


Figure 31: Monitoring the path probability

Convergence index

For AntNet the reinforcement learning factor r is considered to be an indication of the convergence of the routing solution. If the r is close to r_{max}[17] the current solution is good. Similarly, the temperature γ for CEAnts is an indication of the convergence where the temperature should stabilize when most of the paths found have the same value.

In Figure 32 it is illustrated how the convergence index for AntNet and CEAnts changes as the topology changes.

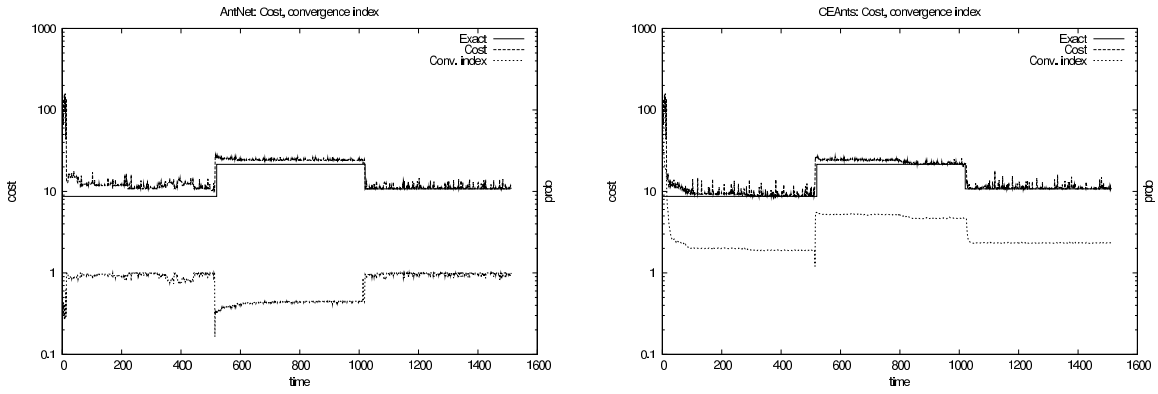


Figure 32: Monitoring the cost value and convergence index - 10 replications

After a short initial exploration phase (almost not visible) the r-factor approaches 1 and the γ stabilizes. When a change in the topology causes the cost to go up, significant changes in the convergence indices are observed. From this it seems that the convergence index, both for AntNet and CEAnts, is a good candidate for a new monitoring index. Through this it can be observed that the network conditions have changed. Observe also that this index is unique for each ant species. Hence, it is possible to follow the changes per route between end nodes s and d. This resembles the well-known active measurement technique using ICMP ping protocol.

[17] In the current study the $r_max = 1$

Cost value index

From Figure 32 it seems that the cost value itself is an alternative to the use of the convergence index (r and γ) for monitoring the network condition. However, due to deviations from the best path, e.g. through the use of exploration ants, the cost value has a tendency to high variation over time, see e.g. the Figures 26 and 33, and the convergence index is some sort of time derivative of the cost value. For AntNet the r is an exponential average mean over the observation window, while the γ in CEAnts is the result from optimizing the change of measure (the change of routing probabilities).

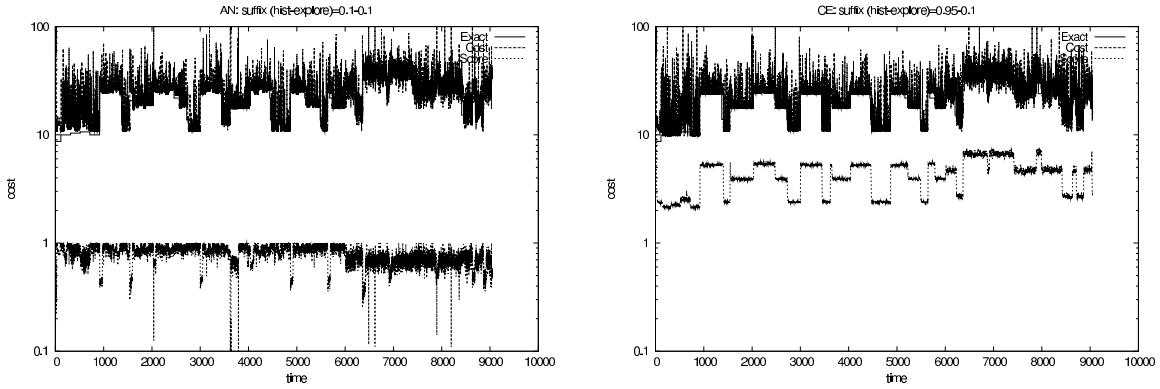


Figure 33: Monitoring the cost value and convergence index - single experiment

In summary, the convergence index of the ant agents are a better monitoring index than the cost values.

Pheromones

Inside the nodes in the network the pheromones are stored. From this the routing probabilities are calculated simply as a normalization of the pheromone values over the available interfaces in a given node.

If the pheromone values are plotted over the simulation periods for each interface of the different nodes, it is possible to observe whether a specific interface of a node is part of any of the best paths found. The interfaces that are part of the current best paths will have high pheromone values, while the other will be low.

In Figure 34 the pheromone values are plotted for 4 different nodes in the network topology from Figure 21. The network topology changes from Phase 1 to 2 at time 500, and from Phase 2 to 3 at time 1000. See Section 7.4 for description of the 3 phases. The pheromone values for all active interfaces for the 4 nodes are plotted, both for AntNet and CEAnts.

It can be observed from Figure 34 that the pheromone values are different for AntNet and CEAnts because the AntNet formulation implies that the pheromone values are normalized and hence equal to the routing probabilities. A more important difference is that AntNet seems to retain a high pheromone value on an interface even if this interface is no longer part of any of the best paths. E.g. interface 1 of node 22 is not part of the best path in phase 1 (time 0 to

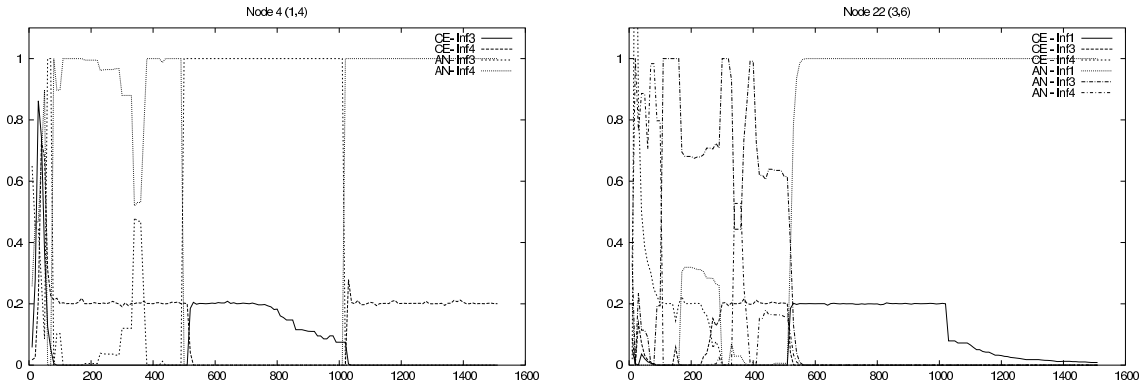


Figure 34: Monitoring the pheromone values of the interfaces in 2 different nodes

500) or phase 3 (time 1000-1500), but is a part of the best path of phase 2 (time 500-1000). The pheromone value of interface 1 of node 22 is increased as expected when entering phase 2, but does not evaporate when leaving phase 2. Hence, the pheromone values of AntNet does not change when the network topology changes unless another interface on the same node become part of the new best path. The pheromone values of CEAnts will evaporate unless they are updated with a strong value. This means that reading pheromone values using CEAnts will give a good indication of whether this is part of one of the best paths, or if it has recently been.

A supplement to observing the pheromone values, it is possible to count the number of agents over each interface. When the number is high, this is a popular interface and part of the current best paths. If the number is low, or zero, this mean that the interface is rarely used.

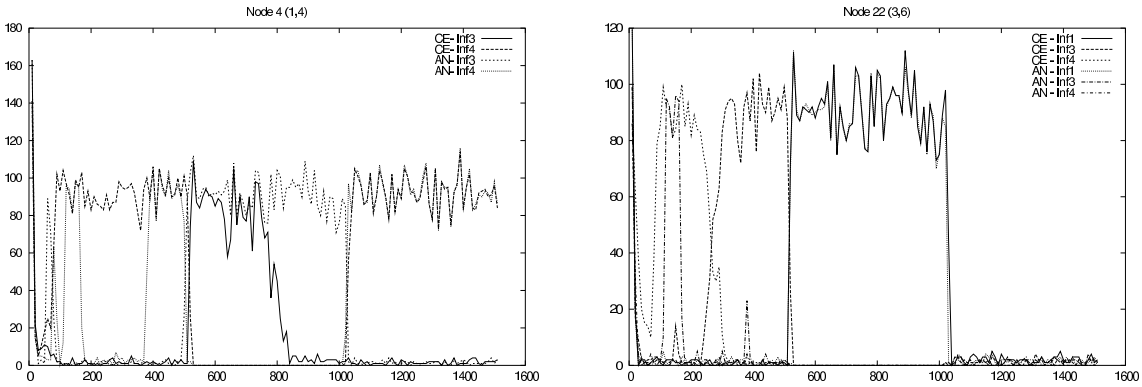


Figure 35: Monitoring the number of agents over the interfaces in 2 different nodes

From Figure 35 it can be observe that the number of agents over a specific interface is clearly varying dependent on whether the interface is part of the best path or not.

In summary, the pheromone values of AntNet does not change when the network topology changes unless another interface on the same node become part of the new best path. The pheromone values of CEAnts will evaporate unless they are updated with a strong value. This

means that reading pheromone values using CEAnts will give a good indication of whether this is part of one of the best paths, or if it has recently been.

For both AntNet and CEAnts, reading the number of agents over each interface will give a clear indication of whether the interface is *currently* part of any of the good paths, or not.

7.6 Concluding remarks

In this section the transient behavior of ant-based routing has been studied. A network is unstable (dynamic) due to changes in topology caused by link and node failures, mis-configurations, and redesign. The data traffic served by this network is very bursty and will influence the routing function when the routing metric is delay and bandwidth sensitive.

Identifying indicator of dynamics or transience in the ant-based routing algorithm will potentially provide more accurate information network state to the network operator. If the ant-based routing algorithm contains information that could improve the global behavior of the network, this should result in rules and algorithms that are instantiated on the nodes e.g. in an ad-hoc network or to a central management or control center in a fixed IP network. This means, for instance, that if a reactive routing approach is taken, see Section 1, the indices of the reactive routing algorithm are observed, and changes in these indices could trigger an alarm and active some local rule, or spread this alarm in the neighborhood by diffusion or using the approach described in Section 6.

Summary of observations

The preliminary results in this section are promising with respect to their abilities to adapt to network changes. CAS indices are identified that can be observed and give helpful input to the monitor of network conditions and and detect changes in these.

AntNet and CEAnts react to changes in topology that both increases and decreases the object function. However, if the memory is too long some of the changes (the cost value improvements) will not be detected before the topology is changed again. It is an obvious challenge to balance the memory factor in order to rapidly detect all topology changes, and at the same time not end up with a solution far from the optimal, or a solution being to instable.

The results reported in this section demonstrate that some of the CAS indices seems to give very useful information about the changes in network conditions. Hence, CAS indices can give helpful input to the monitoring of network conditions and and detect changes in these.

The path probability could be used as an indication of exploration (instability) but it is not evident that it is suitable for detecting changes in the network conditions.

When a change in the topology causes the cost to go up, significant changes in the convergence indices are observed. This index is unique for each ant species. Hence, it is possible to follow the changes per route between end node A and B. This resembles the well-known active measurement technique using ICMP ping protocol. The convergence index of the ant agents are a better monitoring index than the cost values.

The pheromone values of AntNet does not change when the network topology changes unless another interface on the same node become part of the new best path. The pheromone values

of CEAnts will evaporate unless they are updated with a strong value. This means that reading pheromone values using CEAnts will give a good indication of whether this is part of one of the best paths, or if it has recently been.

The CAS indices can be applied as input to traffic engineering. This will be explored further, see Table 2 for a few examples.

Table 2: Examples of use of CAS indices

Metric	Ex. of observation	Ex. of "health"	Ex. of alarm
Ant route table	Deviation from data routing table	Misconfiguration in routing, interface overload	Significant deviation (in time or space)
Pheromone values	Increase by 20% in 1 sec.	Node/link/path down	Check configuration
Convergence index	Decrease by 20% in 1 sec.	New node/link/path discovered	None
Cost value index	Average over 5 sec. decreased by 10% last minute	Aftereffect of change in network (still exploration)	None
Path probability	Close to max. for last minute	Stable network	None

Innovative aspects

Previously proposed ant-based routing methods are compared and applied to a highly, dynamic network environment with frequent changes in topology and traffic load.

It is proposed to use an ant-based routing model for monitoring the changes in network conditions. This can be done even if the routing model is not used for controlling the actual routing in the network itself.

It is considered to be possible to integrate the reading of CAS indices with other network monitoring means. E.g. both observing the pheromones and the number of agents could be included in the Measurement Information Base (MIB) which can be polled by a network manager though Simple Network Management Protocol (SNMP).

Further plans

More experiments will be conducted. Larger topologies with other structures will be studied, i.e. realistic sized, national-wide Internet, with respect to monitoring. Furthermore, a more complex routing problem will be studied, i.e. multi-point routing, and search for primary and secondary paths.

A closer look at the parameters of AntNet and CEAnts in dynamic environments will bring new knowledge on their robustness, and hopefully provide general advise for tuning the parameters.

It is expected that the CAS indices can be applied as input to traffic engineering. This must be explored further, see Table 2 for a few examples.

In addition, the monitoring objectives of the demonstrator defined in WP4 of the BISON project must be clarified. A part of this is to conduct simulation studies of the topology that will be defined in this demo activity.

Conclusions

This document reports on the models for basic services studied in BISON, which are: routing, search, topology management, collective computations and monitoring. Every model is discussed by describing its characteristics and pointing out its analogies with biological systems and its relationships with other related work in the field. For some models some preliminary results were already available and have been shortly discussed. For what concerns the bio-inspired background, the models presented here have been designed utilizing notions from ant colonies (routing and monitoring in MANETs), immune system (search in P2P networks) and epidemics (topology management, collective computations and monitoring in overlay networks).

Most of the work described in this deliverable is still ongoing, and we plan for the next months to improve, implement and evaluate our models. Work progress will be reported in deliverable D06 and D07 (both due by month 24) which will discuss respectively implementation and evaluation aspects for the models for basic functions.

A Review of related work on routing

A.1 Related work on routing in MANETs

In this appendix we give a short description of the most important algorithms and research directions for routing in MANETs. The description is organized in paragraphs, each describing a different set of algorithms sharing the same basic mechanism.

Proactive protocols

One of the first routing algorithms for MANETs was *Destination-Sequenced Distance-Vector Routing* (DSDV). It was published in [121] in 1994, and is an adaptation of the classical Bellman-Ford routing algorithm (e.g., [13]. It is a proactive routing protocol, which means that nodes keep paths to all other nodes, also the ones they are not currently communicating with. More precisely, every node keeps a distance vector table, indicating for each destination the distance (in number of hops) and the next hop. Each node monitors its own local connections, and periodically broadcasts updates of its routing table over the network. Nodes adapt their routing tables according to their own observations and the updates received from other nodes. The main difference with traditional distance-vector algorithms is that routes have sequence numbers assigned to them by the destination, so that nodes can distinguish between different routes and choose the most recent one. This is important in dynamic environments like MANETs, since routing information ages very quickly, and the combined use of old and new information might cause loop formation. The most important drawback of the algorithm is that as the network becomes more dynamic and larger, more routing table updates are broadcasted by the nodes, giving rise to consistent overhead.

Reactive protocols

A lot of the overhead which reduces the performance of proactive protocols can be avoided if nodes only keep routes for destinations they are actually sending data to. This is the idea behind reactive routing: nodes do not gather routing information unless they have data to send. Then they set up a route on-demand for the duration of the session. The most important reactive algorithms are *Ad-Hoc On-Demand Distance Vector Routing* (AODV) [122] and *Dynamic Source Routing* (DSR) [90]. In AODV, a node which has data to send will flood a route request message (RREQ) through the network. When the RREQ reaches the destination, a route reply message (RREP) is generated which travels back to the source. On its way to the source, the RREP sets in intermediate nodes next-hop pointers indicating the route to the destination. These pointers are then used by the data. When a connection on the route breaks (two neighbours on the route move too far apart), a route error message (RERR) is sent back to the source, and the source starts a new RREQ. The strategy of DSR is quite similar, the main difference being that the algorithm does not set up next-hop pointers to indicate the path to the destination, but instead brings the full path back to the source. This full path is then attached to each data packet (resulting in *source routing*). One of the advantages of this approach is that nodes can extract paths from data packets and use them to build a route cache (packet overhearing) The availability of cached routes can speed up route setup (an important weak point of reactive

protocols). A clear disadvantage is that there is extra overhead for each data packet. It is not completely clear which one of AODV or DSR is better. Comparative studies [20, 30, 31] agree though that AODV and DSR perform better than DSDV.

Mixed protocols

The algorithms described so far are among the most cited in the area. However, a whole lot of other algorithms have been proposed, many of which with interesting features and good performance. An interesting class of algorithms are the *mixed protocols* [64], which adopt aspects of both proactive and reactive routing. An excellent example is *Zone Routing Protocol* (ZRP) [75]: proactive routing is used within a certain zone around each node (so each node proactively keeps up-to-date routes to nodes in a two or three hop radius), and reactive routing is used for destinations further away. An interesting effect is that reactive route setups can go quite fast, as route requests can travel immediately towards to borders of the proactive zones, rather than travel by blind flooding.

Position based protocols

Thanks to the increasing availability of small, cheap GPS receivers, it makes sense to create algorithms which assume that nodes know their own *geographic coordinates*. The most simple scheme is then *greedy forwarding*, in which a data packet is forwarded to the neighbour which is closest to the destination (see e.g. [144, 58]). In these simple algorithms, it is sufficient to obtain the destination's location and the neighbours' locations to be able to perform routing. The advantage of location-based algorithms is that the system can scale well, because routing control messages can stay very limited and simple, and routes can be flexible (unlike in DSR or AODV for example, where a path consists of a fixed list of nodes). Disadvantages are the dependence on GPS and the problem of how to find out the position of the destination node (some possible solutions to this last problem are presented in [102] and in the work of the *Terminodes* project [14, 16]. Another problem is the fact that these algorithms can experience difficulties when there is a big connectivity gap in the graph, because then greedy forwarding does not find the target. Some examples of possible solutions to this problem are given in [99] and in [15]. A good overview of existing position based algorithms can be found in [65].

Partitioning and neighbour selection protocols

One can distinguish among MANET routing algorithms based on the way they structure the network. DSDV, AODV and DSR are *uniform protocols*, which treat all nodes as equal, and work in a flat network structure. Partitioning algorithms, on the other hand, try to create a *hierarchy* in the network which allows them to obtain routes more efficiently. Examples are *cluster-based* algorithms like *Clusterhead Gateway Switched Routing* (CGSR) [25]. CGSR divides the MANET into clusters and assigns clusterhead and gateway nodes. Routes are obtained via these clusterheads and gateways. This mechanism leads to faster route discovery, but has as an important drawback the overhead created by the partitioning algorithm (the hierarchy will have to be reconsidered quite often). This prevents this group of algorithms from scaling well.

A better solution for large networks seems to be the group of *neighbour selection* algorithms. In these protocols, each node makes its own decisions as to which other nodes are considered near or far. So, like in partitioning protocols, there is some structure in the network, which makes routing over large distances easier, but there is no overall structure that has to be maintained; the nodes each make a hierarchy from their own point of view. An example is the previously mentioned ZRP, in which a proactive protocol is used for nodes within a certain number of hops and a reactive protocol for nodes further away. Another example is the *friends* mechanism in Terminodes routing [15]. In this mechanism, each node proactively maintains paths to a number of "friend" nodes. When it then searches a path to a destination, it sends a request to its friends, which forward the request to their friends, until there is a node which knows where to find the destination. The friends are chosen in such a way that the friends graph is a *small world graph* (e.g., [9]). In this way, the friend relations between nodes give rise to a small world *overlay graph* which is used in turn to find paths.

Other classes of protocols

Other interesting characteristics one can use to classify MANET routing algorithms are the *cost function*, the *size* of the network considered, and whether *single path* or *multipath* routing is used. Most traditional MANET routing protocols evaluate routes based on the hop count, or on the recency of the route. However, other cost functions, like route delay, could also be used. In *Associativity Based Routing* [147], nodes periodically send associativity messages to their neighbours, and the total count of these messages is a measure for the stability of a link. The algorithm prefers routes with links with high associativity counts. In *Minimum Power Routing* [139], the algorithm prefers the routes with minimum power usage. As for the size of the network, most algorithms are developed for small network sizes (typically 50 to 100 nodes). Only a few algorithms aim for larger scale networks (e.g. Terminodes [14, 15, 16] and MABR [78]). It is not clear however to what extent these algorithms manage to scale better than other algorithms. Finally, most algorithms use single path routing: at any moment in time, the algorithm uses exactly one path between source and destination. In multipath routing, there are several paths between any source and destination, and they are used concurrently. Advantages are increased throughput and more robust behaviour in case of a link failure. The best known multipath MANET routing algorithm is probably *Temporally Ordered Routing Algorithm* (TORA, [120]), which however scores quite badly in comparative studies [20]. In [117], multipath versions of DSR were shown to perform significantly better than the original algorithm. Another multipath routing example is presented in [149]. Finally, also in the previously mentioned Terminodes project multipath routing is used.

A.2 Related work on ant-based routing

AntNet

In AntNet, every node in the network keeps two data structures: a routing table and a local traffic statistics table. The routing table is organized like a distance vector table, with for every destination one entry per outgoing link. The entries are probabilities that play the role of *pheromone*: the probability P_{ij}^x from i's routing table indicates the goodness of sending a message with destination x from i over outgoing link j. The local traffic statistics table keeps a

moving average of the estimated mean and variance of the trip time to each destination. It also keeps the best trip time experienced over a moving observation window. The traffic statistics are updated using the traveling times experienced by ants, and are used to evaluate the trip time of new routes.

Every node s in the network sends *forward ants* out at regular intervals, to randomly chosen destinations. The probability of choosing a destination d depends on the amount of data traffic which is normally generated in the node for destination d (so that for often needed destinations better routing information is available). This does not mean that AntNet is a reactive protocol: ants are sent constantly, and whether or not a path to a certain destination is needed at a specific moment does not have any influence. While travelling to d, a forward ant will record the times which elapse going from node to node until the destination. As the forward ants are placed in the same queues as data packets, these time recordings will be similar to the delays experienced by data packets. The route chosen by the ant is based on the probabilities in the routing tables, adjusted with an estimate of local traffic. This estimate is found in the queue lengths for the different outgoing links. By the use of probabilistic routing, different paths are explored. Once an ant reaches its destination, it turns around and becomes a *backward ant* which returns to the source node s. Backward ants do not use the same queues as data packets, but high priority ones (in order to distribute the new information quicker), and do not follow probabilistic routing (they just follow the exact reverse path of the forward ants).

When a backward ant arrives in a node k on its way back to s, it updates the entries in k's data tables for destination d (with some caution, also entries for nodes between k and d can be updated). First, the traffic statistics table is updated using the trip time experienced by the ant. Then the probability table entry P_{kl}^d is updated, with l being the neighbor over which the ant arrived back at k. The traffic statistics are used to evaluate the quality of the new route from k to d over l, and the probability is adjusted according to this quality. It is possible, however, that there was a major change in the network traffic situation, which would cause the trip time from k to d over any route to change drastically. This could mean that even the optimal route between the two nodes has a trip time which is much higher than the previous ones recorded in the traffic statistics table. To make sure that in this case there would still be an enforcement of the best route, AntNet always gives a small positive increase to P_{kl}^d when an ant comes back to k over l, even when the trip time is much worse than expected. The idea is that the frequency of ants' arrivals will be higher for the best path (because they can come back quicker), and therefore the best path will still get more enforcement.

When data packets have to be sent, the probabilities in the routing table are used. So just like ants, data packets will be routed over multiple paths, according to the estimated goodness of the paths and the local traffic estimates. However, for data packets, the probabilities are raised to a power higher than 1, so that there is a higher preference for paths which were experienced to be good. This is to avoid that too many data packets would follow lowest probability links.

In [38] an improved AntNet model is described. Forward ants do not anymore use data queues but the high priority ones. In this way they can quicker reach their destination. The ant traveling time is estimated directly by the backward ant looking at the queue length of the traversed nodes. This version performs better, especially in the case of large networks, when the paths get longer.

In recent and ongoing work, *AntNet-SELA* [39] and *AntNet++* [36] are presented. Both are

extensions of the original AntNet approach. The main difference with the algorithm described above is that ants are no longer used in a purely proactive way. The new versions of AntNet are mixed protocols: apart from the proactive ants which are sent out at regular intervals, there are also reactive ants. These are sent out when a new communication session is started, and during the whole lifespan of the session, or when special events occur, like a sudden change in the goodness of a certain route. In this way the algorithm can be more adaptive and can obtain better information about the network situation when this is necessary. Other features of AntNet++ include the fact that forward ants can fork when in a node several good possible routes exist, and that the rate at which proactive ants (the traditional ants like they are used in AntNet) are sent out can be adapted.

It is clear that AntNet (and its successors) has some nice features which would be useful in MANETs: it is robust, distributed and highly adaptive. One important problem, however, is the high amount of overhead. The continuous generation of small ant packets to explore the network can lead to heavy congestion in a MANET. Therefore a direct implementation of the algorithm on MANETs does not yield good results (in [10], a quite literal mapping of AntNet onto MANETs is tested with bad results, and in [60] AntNet has a worse performance than the authors' new algorithm). It is clear that the original design of AntNet, which was optimized for wired networks, has to be rethought in order to make it suitable for a MANET environment.

Ants Routing and Accelerated Ants Routing

Some of the first work on ant based routing for dynamic networks is *Ants Routing* [140]. It was not really developed for MANETs though, but for wired networks which are dynamic due to frequent link and node failures. Its successor however, *Accelerated Ants Routing* [109], was really developed and tested for MANETs.

In Ants Routing all nodes keep routing tables which give a probability of using neighbor y for sending to destination x, just like in AntNet. Periodically, every node x sends an ant to a chosen destination d. As this ant passes node y on its way to the destination, y increases its probability for sending to x over the neighbor node the ant came from. The amount of increase depends on the cost the ant experienced while travelling from x to y. An important difference with AntNet is that the ant does not return (there is no backward ant): while the ant travels in one direction, nodes learn about the opposite direction (backward exploration). This is only possible if a symmetric link cost metric is used (like e.g. the number of hops), and is the same mechanisms used in ABC. There are two kinds of ant forwarding mechanisms: regular ants, which follow the probability distribution from the routing table, and uniform ants, which choose a neighbor according to the uniform distribution. The latter are used to enhance exploration and avoid convergence. Data packets always follow the probabilistic routing table. In Accelerated Ants Routing, two simple improvements are introduced. First there is the no-return rule, simply stating that ants choosing a next neighbor will never choose to return to the node they came from. Second, every ant keeps a history list of the N last nodes it passed. In this way, not only the route to the source node can be updated, but also the route to intermediate nodes. The authors test their algorithm on a MANET simulator and find that it converges faster than AntNet.

Ants Routing and Accelerated Ants Routing have some important drawbacks. They work in a table-driven way, so memory requirements and routing overhead will get large as the network

gets bigger. Moreover, the information a node has depends on the ants it has been visited by. If no ants have recently reached node x from node y, x will not have a good route available to y. This is a direct consequence of the backward exploration: route exploration is performed by the ants which are started from the destination, and there is no way that source nodes can start a process to find a certain destination.

PERA

In [10], two MANET routing algorithms are proposed. The first is very similar to AntNet, the main differences being that the forward ants are sometimes routed uniformly rather than according to the routing probabilities (to enhance exploration), and that data are routed *deterministically* over the highest probability path. The authors find that this algorithm does not work very well, due to the large amount of routing overhead and the inefficient route discovery. Then they propose a second algorithm, which they call *Probabilistic Emergent Routing Algorithm* (PERA). The main difference with the first algorithm is that it is reactive: forward ants are only sent out when a route is needed. Also, they are broadcasted instead of unicasted to the destination.[18] In this way, the forward ants are in fact very similar to route request messages in AODV and DSR. The difference with traditional route discovery is that different route discovery agents reaching the destination give rise to different route replies (backward ants) and therefore multiple routes between source and destination (in DSR and AODV only one route is set up). This makes it possible to use multiple routes for the data packets. The authors mention this possibility, but from the paper it is not clear whether they actually do this or just route the data packets over the route with highest probability. In any case, the adapted algorithm is more or less a multipath version of AODV. Experimental results show that the performance of the algorithm is quite comparable to that of AODV.

ARA

Ant-Colony-Based Routing Algorithm (ARA) [71, 72] is another ant-based routing algorithm which does not differ much from AODV and DSR. It works in an *on-demand* way, with nodes sending forward ants out to destinations with which communication is needed, and backward ants coming back to them. Unlike in AntNet, backward ants are not used to report back the trip times experienced by forward ants: both forward ants and backward ants will leave information about the path they are coming from. So forward ants leave pheromone trail indicating how to reach the source, and backward ants leave pheromone about how to reach the destination. In this way an on-demand path is set up. Because of the use of ants, multiple paths will be set up, which can be used for backup in case of route failure. An interesting feature is the fact that ordinary data packets will reinforce pheromone both in the direction they are coming from and in the direction they are going in. This approach allows to use less ants, as the data packets take over a task which would normally be performed by ants (in fact, it is like piggy-backing an ant onto each data packet). The ARA algorithm was compared to AODV and DSR and found to perform better than AODV but slightly worse than DSR in highly dynamic environments. Interesting though is that it scores pretty well in terms of routing overhead.

[18] Broadcasting is more efficient as it takes advantage of the inherent broadcast nature of antenna communication. Also, it creates less overhead at the MAC layer, as the normal RTS/CTS handshake before transmission and the ACK message after transmission are not used.

Termite

In *Termite* routing [128], each node keeps a pheromone table with one entry per known destination and per neighbour. Data packets are routed probabilistically according to this table. When a data packet arrives at a node which does not have pheromone for its destination, route request packets (RREQ) are launched. They do a *random walk* until they find a node which does have pheromone for the destination. Once an RREQ arrives at such a node, it will send a route reply back to the source (RREP), which will extend the pheromone trail found in the node until the source which needed the trail. Apart from the RREP and RREQ packets, nodes can also send HELLO packets (to find out about their neighbours), and SEED packets (which do a random walk to spread pheromone). In this system, pheromone is increased by every kind of packet (also data packets): at the arrival of a packet, pheromone indicating the path to the source of the packet is increased. Pheromone increase is a fixed value, independent from the path quality: the authors expect that bad paths will automatically transport less packets and will, thanks to the periodic pheromone decay, eventually be eliminated. This algorithm is supposed to have low overhead and complexity. A negative point is that it often produces sub-optimal paths. It has not been compared to AODV or DSR.

MABR

Mobile Ant Based Routing (MABR) [78] aims at WAN MANETs. It is a rather complicated algorithm. The whole area of the MANET gets divided into zones (these are rectangular zones bound to a fixed location). On the smallest level, all the nodes of one zone together make up a logical router. A logical router maintains a logical link to every other zone, which is just a set of coordinates close to the geographical center of that zone. When a message is sent over a logical link, it is forwarded to the neighbor closest to the link's endpoint (with possibly some extra mechanism to go around small gaps in the network). To go to a certain destination zone, it might be possible to go over the logical link leading straight to that zone, or it could instead be necessary to follow a combination of other logical links (e.g., if there is a big gap in the network between the two zones). Ants are used to test this: they are proactively routed to the different destinations, and they bring back trip times. These trip times are used to update probabilistic routing tables. The whole MABR project has only just started, and no test results are available yet. Some implementation problems could be hard to solve: e.g., how a node finds out in which zone its destination is, and how routing information can be distributed efficiently among the nodes of a zone.

Other algorithms

In [21], the authors describe a *location based* algorithm for MANETs which uses ants to spread the routing information. Every node keeps in a routing table location information of all the other nodes, and routing paths are calculated with a shortest path algorithm. To keep the tables up-to-date, information is exchanged locally among neighbors, and globally by sending ants to nodes further away. So, in this approach, ants replace flooding as a way to spread routing information over the network. The approach is evaluated in a simple simulation study, and is claimed to cause less overhead than another location based routing algorithm which uses a

form of restricted flooding. It seems hard to believe that this approach would scale very well, because every node has to keep track of the position of all other nodes.

In [114] a number of mobile agents go around the MANET, independently from the nodes: they are not generated by the nodes, and they do not die at a certain destination, they just keep going round. They keep a history list of their N last visited nodes, and update the nodes' routing tables with information derived from this history list. At any node, the agents choose the least recently visited neighbor as their next hop. In simulations, the algorithm managed to provide paths between nodes in the network, but probably not enough to be a really good performing routing algorithm.

A quite simple hybrid approach, called *Ant-AODV*, was proposed in [108]. It is in fact a combination of the approach just described and AODV. A certain number of ants keeps going round the MANET in a more or less random manner, keeping track of the last N nodes they visited. When an ant visits a node, it will update the node's routing tables. Then, when a message has to be sent, the node will first check whether it has a fresh enough route available. If it does not, it will launch a route discovery process identical to the one in AODV. The use of ants is supposed to speed up AODV in three ways. First, there is a better chance that the node will have a recent route available. Second, if it doesn't, it will still get a faster reply to its route request message as there is a better chance that a nearby node knows a recent route. Finally, while a node is sending over a certain path, it can be informed by an ant of a better path, and it can switch to this one. According to the simulation studies in this paper, the Ant-AODV approach performs better than Ant-based routing or AODV separately. It could be interesting to compare this approach to DSR since extracting new routes from packets passing by is in a sense similar to the caching mechanism used in DSR.

B Review of related work on performance monitoring

This appendix gives a review of the state-of-the-art in performance monitoring relevant for the BISON project. The scope of the monitoring function is in both P2P and mobile ad hoc networks.

What do we mean by performance monitoring?

In general; monitoring is supervision of resource usage, utilization and availability. Examples of resources in BISON context are information content (documents, video, music, etc.), processing power, link capacities, routers and terminals, battery power. Monitoring is a general term also applied in many other fields, e.g. a general definition is found in the field of environmental protection [157], where monitoring is defined *as the periodic oversight of a process, or the implementation of an activity, which seeks to establish the extent to which input deliveries, work schedules, other required actions and targeted outputs are proceeding according to plan, so that timely action can be taken to correct the deficiencies detected*. The *processes* is in BISON context e.g. the traffic profiles, the service usage, server, scheduling, load sharing. The *timely actions* are network management, and *deficiencies* are traffic overload, route flaps, link/router/server down, etc.

More specifically; network and service monitoring is the oversight or supervision of the resources relevant for BISON. The *performance* monitoring function is relevant and important both from a user and provider perspective. The user (consumer) is interested in monitoring the quality of the delivered service, while the provider is concerned with maximizing the resource utilization when delivering service within the given quality guarantees. The service quality constraints are part of a Service Level Agreement (SLA) between consumer and provider. In the SLA, the monitoring function should also be described in order for the consumer and provider to have common understanding of what the service quality is and how this should be verified through measurements.

Monitoring objectives in BISON is considered to be:

- Service dependability (availability and reliability) in ad-hoc and P2P networks
- Performance and utilization of network elements (ad-hoc) and service platforms (P2P)

BISON consider monitoring in both user and provider perspective.

It is important to emphasis that performance monitoring is NOT network monitoring in the sense of surveillance of the content of communications and individuals' network usage.

In the following monitoring stands for performance monitoring.

Monitoring in packet-switched networks

There are mainly two approaches for monitoring of networks and service platforms; *Active monitoring* where test packets are submitted and their response (status, delay, route, etc.) are observed, and *Passive monitoring* where server logs and packet counters are sampled, or packets are (partially) captured.

A lot of work is being done in the field of monitoring. For the BISON project the particular interest is the monitoring and measurement techniques for IP based (i.e. packet) network, see for example [19] for state-of-art and [69] for an excellent tutorial in IP monitoring. The tutorial includes:

1. **Active monitoring**: Active means sending test traffic/packets and observing the performance of this traffic/packets. In IP networks active techniques are in widespread use both on application, transport and network layers. Examples are use of built-in functions on end-user equipment (ICMP ping, trace-route) and dedicated routers (RIPE NCC) or route functions (Service Assurance Agents of the Cisco IOS).
2. **Passive monitoring**: Passive means observing user traffic by counters or traffic/packet capturing. Sampling of interface counters on routers (e.g. by SNMP), reading server logs, packet capturing by sniffers are examples of passive techniques. Major challenges include reducing and accumulating data without throwing away valuable information input to traffic engineering, accounting, security surveillance, etc. A huge number of both commercial and free products are available.

There has also been done work on combined active and passive techniques, e.g. ATM Forum has defined OAM cells to be sent periodically (actively) dependent on the traffic load (observed passively). A similar approach is proposed by Lindh [103] for IP networks but not yet adopted by the IETF.

Performance monitoring and measurement techniques are part of the work in the standardization bodies. Both Internet Engineering Task Force, IETF (www.ietf.org) and ITU-T (www.itu.org) address these issues.

In IETF performance issues are part of several working groups (WGs):

- Transport Area:
  - **ippm** - IP Performance Metrics WG will develop a set of standard metrics that can be applied to the quality, performance, and reliability of Internet data delivery services
  - **rtfm** - Real-time Traffic Flow Measurement WG (now concluded) was responsible for issues in traffic flow measurement, and developed a Traffic Flow Model (implemented in NeTraMet [118], and measurement architecture including Meter MIB.
- Operations and Management Area:
  - **SNMP** - Simple Network Management Protocol is used for configuration and monitoring network status. The SNMP is the responsibility of several groups.
  - **psamp** - Packet Sampling WG will define a standard set of capabilities for network elements to sample. subsets of packets by statistical and other methods
  - **rmonmib** - Remote Network Monitoring WG will define a set of managed objects for remote monitoring of networks. Other Measurement Information Bases (MIBs) are proposed by other working groups in the Operation and Management Area.

In ITU-T, in particular the study groups 2, 4, 12 and 13 deals with monitoring and measurement issues.

- Study Group 2 (Operational aspects of service provision, networks and performance) are responsible for several studies including operational aspects of inter-working between traditional telecommunication networks and evolving networks;
- Study Group 4 (Telecommunication management, including TMN) are responsible for studies regarding the management of telecommunication services, networks, and equipment in addition to transport-related operations procedures, and test and measurement techniques and instrumentation.
- Study Group 12 (End-to-end transmission performance of networks and terminals) are responsible for guidance on the end-to-end transmission performance of networks, terminals and their interactions, in relation to the perceived quality and acceptance However, this monitoring is not monitoring the P2P application. by users of text, speech, and image applications.
- Study Group 13 (Multi-protocol and IP-based networks and their inter-networking) is the lead Study Group for IP related matters, B-ISDN, Global Information Infrastructure and satellite matters. They are responsible for the I-series and in particular *I.380 Internet protocol data communication service - IP packet transfer and availability performance parameters* which defines performance parameters describing the speed, accuracy, dependability and availability of IP packet transfer.

Monitoring in AHN and P2P networks

Monitoring of Ad-hoc networks can serve two purposes:

- **Ad-hoc Network view**: monitor traffic load, traffic patterns, terminal availability/health in AHN domain to enable load sharing and optimized routing. If no central control center exits (typically not) this information, or at least parts of it, must be known to the terminals.
- **Terminal/User view**: monitor power consumption, coverage (terminal neighborhood), pass-through data and signaling (control plane) traffic, service availability.

Performance monitoring techniques for ad-hoc networks has not been a major issue so far. The reasons is probably that there are few AHN with central control and with critical application and commercial interests. One exception is the use of wireless sensors (low-cost, low-performance entities) connected in ad-hoc network. They can be use e.g. for fire-alarms. The scarce resource is battery power [83]. The objective of the monitoring function is to provide information about the availability (existence/health) of the sensors (on low battery the sensor is unavailable or dead) to a control center. The performance measures are probability of false alarms and the response times. The approach is typically active to detect whether a sensor is alive or not.

Similar to AHN, monitoring of P2P network has two objectives:

- **P2P network view**: global performance monitoring similar to IP network monitoring, i.e. read status on access links, memory usage, disc access, etc. New P2P applications for enterprise will include monitoring hooks for traffic engineering, e.g. JXTA [93] includes peer-monitoring hooks that will enable the management of a peer node.
- **P2P terminal view**: consider access rate (fixed), round-trip time (active monitoring) and file transfer rate or throughput (passive monitoring) for selection of best server/content/peering provider.

As distributed or grid computing and P2P application/information sharing have gained increasing interest, the need for monitoring the performance is increasingly important. For distributed or grid computing the monitoring objective is typically a network view; how to optimism the load sharing and how to control or at least known the performance. The migration from monolithic computing systems to parallel and distributed computing will obviously increase your processing power but at the same time decrease the control and knowledge of the performance of your applications and services. New monitoring tools are under development. For example, in [110] a new tool for monitoring distributed application behavior is presented. The tool is called ANSAmon and is in itself a distributed processing system. In order to simplify management of distributed monitoring processes, a tool that is embedded in middle-ware might be the correct direction, e.g. as described in [113].

From a users point of view it is interesting to know performance attributes related to the different peering points where the requested information or application is available. For example, the access rate (bit/s), the response time (round-trip time which include the peering point response), the available bandwidth (like knowing the received throughput in advance). The most popular P2P applications today have very limited support for monitoring.

Another aspect of performance monitoring in relation to P2P is the performance monitoring of the underlying network [19]. Performance studies has discovered that the web is no longer always the dominant Internet traffic. E.g. see Figure 36 showing that P2P applications (Kazaa and DirectConnect) is the 2 dominant applications on the Internet access between the NTNU and Uninett (the organization responsible for the non-commercial part of the Internet in Norway). The Web traffic is pushed down to a 3rd place. This is important input to the network manager and owner when reviewing their management policies.

Monitoring using CAS

To the author's knowledge there are no published work on using CAS for monitoring a network. The closest we get is monitoring of CAS, which is the other way around.

On the use of CAS for monitoring

The idea behind looking at CAS for monitoring is two-fold

[19] Recall that this does not mean surveillance of content or individual usage

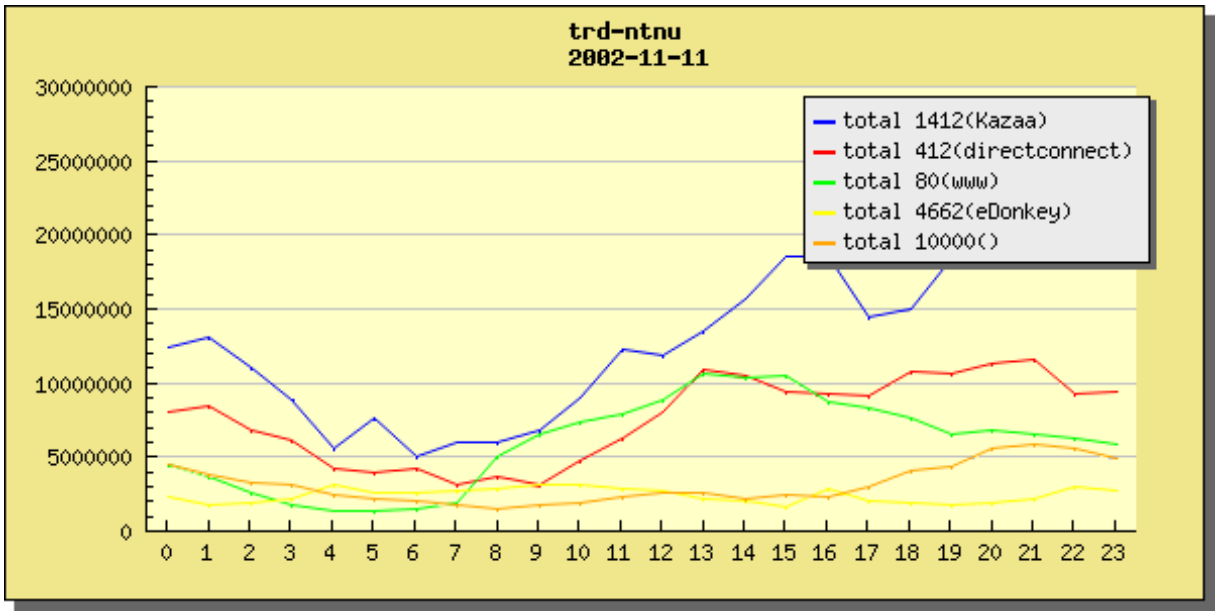


Figure 36: P2P applications dominates the Internet traffic

- It will (potentially) provide essential global information stored locally.

- It is easier for operational personnel to accept CAS for monitoring than CAS for automatic control.

In the following of this section a few issues regarding the use of CAS for performance monitoring in IP based network, and more specifically, in P2P and ad hoc networks, are discussed.

Swarm intelligence, ants in specific, has been proposed to create an adaptive and robust routing function, see "SoA Routing" section for more details. The CAS algorithm is valuable even if the network operator is reluctant to let the routing of the data traffic use the ant routing table. The ant routing table with its pheromons can be considered as snapshot/sample of the current network state. This network state reflects both traffic load and traffic matrix, network capacities, and availability of the network elements.

Using CAS monitoring might be considered as a combination of active and passive techniques where the ants are active test packets exploring the network and looking for the best routes (best in accordance to some quality attributes like the number of hops, total delay, available bandwidth, etc.) and where the ant routing tables and pheromons are passively sampled by some control center.

In Figure 37 an illustration of how CAS designed for routing might serve as input to monitoring function. The pheronoms in the ant routing table is updated, the "temperature" indicate the grade of convergence (the strengths of the signal). The control center is polling the temperature and receives an alarm then a rapid increase in the grade of converege is observed.

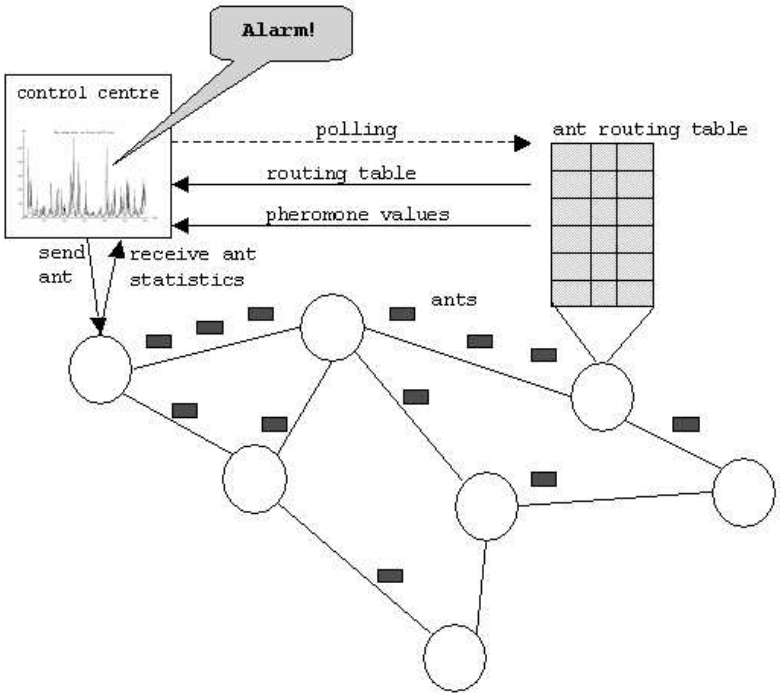


Figure 37: CAS for monitoring

Application in the BISON's context

In general, the hypothesis is that independent of the function the CAS algorithm is designed for, the pheromon "signals" have strong link to the network state and are applicable as indication of the performance and health of the network.

This means that:

- The CAS algorithms for "discovery", "routing" and "routing+search" functions in AHN might provide information in terminal i about the state of the network observable from this terminal i, i.e. all terminals j that have at least one observe ant route in common.
- The CAS algorithms for "search" and "distributed (grid) processing" function in P2P might provide global information locally.

Application in more general contexts

As stated in previous section, the hypothesis is that any CAS algorithm will contain information that can be applied for performance and network health monitoring. This means that in any network where it is possible to implement and execute a CAS, new and valuable information about network state is available that is important input to e.g. traffic engineering.

References

[1] T. Abe, S. A. Levin, and M. Higashi, editors. *Biodiversity: an ecological perspective*. Springer-Verlag, New York, USA, 1997.

[2] Arup Acharya, Archan Misra, and Sorav Bansal. A label-switching packet forwarding architecture for multi-hop wireless LANs. In *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, 2002.

[3] Lada A. Adamic and Bernardo A. Huberman. Zipf's law and the Internet. *Glottometrics*, 3:143–150, 2002.

[4] E. Althaus, G. Călinescu, I.I. Măndoiu, S. Prasad, N. Tchervenski, and A. Zelikovsky. Power efficient range assignment in ad-hoc wireless networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, pages 1889–1894, 2003.

[5] Norman T. J. Bailey. *The mathematical theory of infectious diseases and its applications*. Griffin, London, second edition, 1975.

[6] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of ACM SIGCOMM '96*, August 1996.

[7] T. R. Balch. Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots*, 8(3):209–238, 2000.

[8] Albert-László Barabási. *Linked: the new science of networks*. Perseus, Cambridge, Mass., 2002.

[9] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[10] John S. Baras and Harsh Mehta. A probabilistic emergent routing algorithm for mobile ad hoc networks. In *WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[11] Mayank Bawa, Hector Garcia-Molina, Aristides Gionis, and Rajeev Motwani. Estimating aggregates on a peer-to-peer network. submitted for publication.

[12] H. Bersini and F. J. Varela. The Immune Recruitment Mechanism: A Selective Evolutionary Strategy. In 4^{th} *International Conference on Genetic Algorithms*, pages 520–526, 1991.

[13] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[14] Ljubica Blazevic, Levente Buttyan, Srdan Capkun, Silvia Giordano, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 39(6), June 2001.

[15] Ljubica Blazevic, Silvia Giordano, and Jean-Yves Le Boudec. Anchored path discovery in terminode routing. In *Proceedings of The Second IFIP-TC6 Networking Conference (Networking 2002)*, May 2002.

[16] Ljubica Blazevic, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. *Journal of Cluster Computing*, April 2002.

[17] M. Bodlaender, J. Guidi, and L. Heerink. Enhancing UPnP Discovery with Liveness. In *Consumer Communications and Networking Conference*, Los Alamitos, CA., January 2004. IEEE Computer Society Press.

[18] Béla Bollobás. *Random graphs*. Cambridge University Press, Cambridge; New York, second edition, 2001.

[19] Tønnes Brekne, Marius Clemetsen, Poul E. Heegaard, Tone Ingvaldsen, and Brynjar Viken. State of the Art in Performance Monitoring. Technical Report R 15/2002, Telenor R&D, 2002.

[20] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98)*, 1998.

[21] Daniel Camara and Antonio A. F. Loureiro. A novel routing algorithm for ad hoc networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[22] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2001.

[23] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[24] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8):100–110, October 2002.

[25] C.-C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE SICON'97*, pages 197–211, April 1997.

[26] Cisco. Service assurance agent. Available online at `http://www.cisco.com/en/US/tech/tk447/tk823/tech_protocol_family_home.html`.

[27] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems, 2002.

[28] A.K. Das, R.J. Marks II, M. El-Sharkawi, P. Arabshahi, and A. Gray. The minimum power broadcast problem in wireless networks: an ant colony system approach. In *Proceedings of the IEEE Workshop on Wireless Communications and Networking*, 2002.

[29] A.K. Das, R.J. Marks, M. El-Sharkawi, P. Arabshani, and A. Gray. Minimum power broadcast trees for wireless networks: integer programming formulations. In *Proceedings of the IEEE INFOCOM 2003 Conference*, 2003.

[30] Samir R. Das, Robert Castaneda, Jiangtao Yan, and Rimli Sengupta. Comparative performance evaluation of routing protocols for mobile, ad hoc networks. In *International Conference on Computer Communications and Networks*, pages 153–161, 1998.

[31] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications Magazine special issue on Ad hoc Networking*, pages 16–28, February 2001.

[32] D. Dasgupta and N. Attoh-Okine. Immunity-Based Systems: A Survey. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernatics*, pages 363–374, 1997.

[33] D. Dasgupta and S. Forrest. Novelty Detection in Time Series Data using Ideas from Immunology. In *ISCA 5th International Conference on Intelligent Systems*, 1996.

[34] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic Algorithms for Replicated Database Management. In *Proc. of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, August 1987. ACM.

[35] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[36] G. Di Caro. A society of ant-like agents for adaptive routing in networks. DEA thesis in Applied Sciences, Polytechnic School, Universitè Libre de Bruxelles, Brussels (Belgium), May 2001.

[37] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.

[38] G. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*, pages 541–546. IASTED/ACTA Press, 1998.

[39] G. Di Caro and T. Vasilakos. Ant-SELA: Ant-agents and stochastic automata learn adaptive routing tables for QoS routing in ATM networks. *ANTS'2000 - From Ant Colonies to Artificial Ants: Second International Workshop on Ant Colony Optimization*, Brussels (Belgium), September 2000.

[40] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, July 2001.

[41] J.J. Dongarra. Performance of various computers using standard linear algebra software in a fortran environment. Technical Report CS-89-85, University of Tennessee, October 2003.

[42] M. Dorigo, E. Bonabeau, and G. Théraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[43] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.

[44] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.

[45] M. Dorigo, G. Di Caro, and T. Stützle (Editors). Ant algorithms. Special Issue on *Future Generation Computer Systems (FGCS)*, 16(8), 2000.

[46] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[47] M. Dorigo, G. Di Caro, and M. Sampels, editors. *Ant Algorithms - Proceedings of ANTS 2002, Third International Workshop on Ant Algorithms, Brussels, Belgium, September 12–14, 2002*, volume 2463 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[48] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[49] M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[50] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[51] Olivier Dousse, Francois Baccelli, and Patrick Thiran. Impact of interferences on connectivity in ad hoc networks. In *Proceedings of IEEE Infocom 2003*, San Francisco, April 2003.

[52] Olivier Dousse, Patrick Thiran, and Martin Hasler. Connectivity in ad-hoc and hybrid networks. In *Proceedings of IEEE Infocom 2002*, pages 1079–1088, New York, June 2002.

[53] Fred C. Dyer. Spatial cognition and navigation in insects. In L. A. Real, editor, *Behavioral Mechanisms in Evolutionary Ecology*, pages 66–98. University of Chicago Press, Chicago, MI, USA, 1994.

[54] Fred C. Dyer. Spatial memory and navigation by honeybees on the scale of the foraging range. *Journal of Experimental Biology*, 199:147–154, 1996.

[55] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[56] Patrick Th. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. From epidemics to distributed computing. *IEEE Computer*. To appear.

[57] J. H. Fewell. Social insect networks. *Science*, 301(26):1867–1869, September 2003.

[58] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISU/RR-87-180, ISI, March 1987.

[59] Freenet. http://freenet.sourceforge.net/.

[60] Kazuyuki Fujita, Akira Saito, Toshihiro Matsui, and Hiroshi Matsuo. An adaptive ant-based routing algorithm used routing history in dynamic networks. In *4th Asia-Pacific Conference on Simulated Evoluation and Learning*, pages 46–50, 2002.

[61] Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), February 2003.

[62] Buğra Gedik and Ling Liu. PeerCQ: A Decentralized and Self-Con guring Peer-to-Peer Information Monitoring System. In *23rd International Conference on Distributed Computing Systems*, pages 490–499, Los Alamitos, CA., May 2003. IEEE, IEEE Computer Society Press.

[63] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multihop networks. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 41–50, 1999.

[64] Silvia Giordano. Mobile ad-hoc networks. In Ivan Stojmenović, editor, *Handbook of Wireless Networks and Mobile Computing*. John Wiley and Sons, January 2002.

[65] Silvia Giordano, Ivan Stojmenovic, and Ljubica Blazevic. *Ad Hoc Wireless Networking*, chapter Position based routing algorithms for ad hoc networks: A taxonomy. Kluwer, 2003.

[66] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[67] P. P. Grassé. *Les Insects dans Leur Univers*. Ed. du Palais de la découverte, Paris, 1946.

[68] P. P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[69] Mattias Grossglauser and Jennifer Rexford. Traffic Management for Network Operation. Tutorial at Sigcomm 2001, San Diego, CA, USA, September 2001.

[70] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 381–394. ACM, ACM Press, 2003.

[71] Mesut Günes, Udo Sorges, and Imed Bouazizi. ARA - the ant-colony based routing algorithm for manets. In *Proceedings of the 2002 ICPP Workshop on Ad Hoc Networks (IWAHN 2002)*, pages 79–85, August 2002.

[72] Mesut Günes and Otto Spaniol. Routing algorithms for mobile multi-hop ad-hoc networks. In *International Workshop on Next Generation Network Technologies*, October 2002.

[73] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'01)*, Göteborg, Sweden, 2001.

[74] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, March 2000.

[75] Zygmunt J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, 1997.

[76] J. F. Hayes. *Modeling and analysis of communication theory*. Plenum press, New York, USA, 1984.

[77] H.-G. Hegering, S. Abeck, and B. Neumair. *Integrated Management of Networked Systems*. Morgan Kaufman, San Mateo, CA., 1999.

[78] Marc Heissenbüttel and Torsten Braun. Ants-based routing in large scale mobile ad-hoc networks. In *Kommunikation in verteilten Systemen (KiVS03)*, March 2003.

[79] Bjarne E. Helvik and Otto Wittner. Using the Cross Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[80] M. Heusse, D. Snyers, S. Guérin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1(2):237–254, 1998.

[81] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Zhao. Distributed object location in a dynamic network. In *Proceedings of Symposium on Parallel Algorithms and Architectures (SPAA)*, August 2002.

[82] B. Hölldobler and E.O. Wilson. *The Ants*. Springer-Verlag, Berlin, Germany, 1990.

[83] Chih-Fan Hsin and Mingyan Liu. A Distributed Monitoring Mechanism for Wireless Sensor Networks. In *WiSe'02*, Atlanta, Georgia, USA, 2002.

[84] J. E. Hunt and D. E. Cooke. Learning Using an Artificial Immune System. *Journal of Network and Computer Applications*, 19:189–212, 1996.

[85] R.J. Marks II, A.K. Das, M. El-Sharkawi, P. Arabshani, and A. Gray. Minimum power broadcast trees for wireless networks: optimizing using the viability lemma. In *Proceedings of the IEEE Interational Symphosium on Circuits and Systems (ISCAS 2002)*, 2002.

[86] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. Gossip-based unstructured overlay networks: An experimental evaluation. Technical Report UBLCS-2003-15, University of Bologna, Department of Computer Science, Bologna, Italy, December 2003.

[87] Márk Jelasity, Wojtek Kowalczyk, and Maarten van Steen. Newscast Computing. Submitted for publication.

[88] Márk Jelasity and Alberto Montresor. Epidemic-Style Proactive Aggregation in Large Overlay Networks. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, Tokyo, Japan, 2004. To appear.

[89] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. A modular paradigm for building self-organizing peer-to-peer applications. In *Proc. of the 1st Internation Workshop on Engineering Self-Organizing Applications (ESOA'03)*, Melbourne, Australia, 2003.

[90] David B. Johnson and David A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer, 1996.

[91] N. L. Johnson. Developmental insights into evolving systems: Roles of diversity, nonselection, self-organization, symbiosis. In M. Bedau, editor, *Artificial Life VII*. MIT Press, Cambridge, MA, 2000.

[92] N. L. Johnson. Importance of diversity: Reconciling natural selection and noncompetitive processes. In J. L. R. Chandler and G. V. d. Vijer, editors, *Closure: Emergent Organizations and Their Dynamics*, volume 901 of *Annals of the New York Academy of Sciences*, New York, USA, 2000.

[93] Project JXTA. Available online at `http://www.jxta.org/`.

[94] J. O. Kephart. A Biologically Inspired Immune System for Computers. In *Proceeding of Artifical Life*, July 1994.

[95] S. Kirkpatrick, C.D. Gelatt, and M.D. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[96] E. Klarreich. Inspired by Immunity. *Nature*, 415:468 – 470, 2002.

[97] L. Kleinrock. *Queueing Systems, Volume 1: Theory*. John Wiley and Sons, New York, USA, 1975.

[98] J.B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Preceedings of the American Mathematical Society*, 7:48–50, 1956.

[99] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. Technical report, Distributed Computing Group, ETH Zürich, December 2002.

[100] Ching Law and Kai-Yeung Siu. Distributed construction of random expander graphs. In *Proceedings of The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2003)*, San Francisco, California, USA, April 2003.

[101] Sung-Ju Lee, Elizabeth M. Belding-Royer, and Charles E. Perkins. Ad hoc on-demand distance-vector routing scalability. *ACM SIGMOBILE Mobile Computing and Communications Review*, July 2002.

[102] JinYang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.

[103] Thomas Lindh. An architecture for embedded monitoring of QoS parameters in IP based virtual private networks. In *PAM2001 - A workshop on Passive and Active Measurements*, pages 114 – 123, Amsterdam, April 2001.

[104] A. Liotta, G. Pavlou, and G. Knight. Exploiting Agent Mobility for Large Scale Network Monitoring. *IEEE Network*, 16(3), May 2002.

[105] Alexander Loser, Felix Naumann, Wolf Siberski, Wolfgang Nejdl, and Uwe Thaden. Semantic overlay clusters within super-peer networks. In *Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.

[106] Q. Lv, P. Cao, E. Cohen, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of the 16th ACM International Conference on Supercomputing*, June 2002.

[107] T.L. Magnanti and L. Wolsey. Optimal trees. In *Network Models, Handbook in Operations Research and Management Science* (M.O. Ball et al. eds.), volume 7, pages 503–615. North-Holland, 1995.

[108] S. Marwaha, C. K. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In *Proceedings of IEEE Globecom*, 2002.

[109] Hiroshi Matsuo and Kouichi Mori. Accelerated ants routing in dynamic networks. In *2nd International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 333–339, 2001.

[110] J. A. McCann and K. J. Manning. Performance Management Tool for Interoperable Environments, 2001.

[111] Jim McCoy. Mojo nation, 2001.

[112] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *The 2001 International Conference on Machine Learning*, 2001.

[113] B. Meyer, M. Heineken, and C. Popien. Performance analysis of distributed applications with ANSAmon, 1995.

[114] Nelson Minar, Kwindla Hultman Kramer, and Pattie Maes. Cooperating mobile agents for dynamic network routing. In Alex Hayzelden and John Bigham, editors, *Software Agents for Future Communication Systems*, chapter 12. Springer-Verlag, 1999.

[115] R. Montemanni and L.M. Gambardella. A new approach for the minimum power broadcast problem in wireless networks. Submitted, October 2003.

[116] Alberto Montresor, Márk Jelasity, and Ozalp Babaoglu. Robust aggregation protocols for large-scale overlay networks. Technical Report UBLCS-2003-16, University of Bologna, Department of Computer Science, Bologna, Italy, December 2003.

[117] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *Mobile Networks and Applications*, August 2001.

[118] NeTraMet. Available online at `http://www.caida.org/tools/measurement/netramet/`.

[119] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 21(6):995–1002, August 2003.

[120] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, April 1997.

[121] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[122] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[123] Jon Postel. Rfc 792: Internet control message protocol (icmp). IETF, September 1981.

[124] T. Rappaport. *Wireless Communications: Principles and Practices*. Prentice Hall, 1996.

[125] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, 2001.

[126] Ripe ncc hompage. (last visisted 2004-02-27).

[127] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal (Special Issue on Peer-to-Peer Networking)*, 6(1), 2002.

[128] M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. In *The Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.

[129] Antony Rowstron and Peter Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In Rachid Guerraoui, editor, *Middleware 2001*, volume 2218 of *Lecture Notes on Computer Science*, pages 329–350, Berlin, November 2001. Springer-Verlag.

[130] Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1999.

[131] Reuven Y. Rubinstein. Noisy networks. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, chapter Combinatorial Optimization, Cross-Entropy, Ants and Rare Events - Section 7. Kluwer Academic Publishers, 2001.

[132] G. Sakarian and H. Unger. Topology Evolution in p2p Distributed Networks. In *Design, Analysis and Simulation of Distributed System*, pages 12–18, 2003.

[133] Scalable Network Technologies, Inc., Culver City, CA, USA. *Qualnet Simulator, Version 3.6*, 2003. `http://stargate.ornl.gov/trb/tft.html`.

[134] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.

[135] S. Singh and S. Thayer. Immunology directed methods for distributed robotics: A novel, immunity-based architecture for robust control & coordination. In *Proceedings of SPIE: Mobile Robots XVI*, volume 4573, November 2001.

[136] S. Singh and S. Thayer. A foundation for kilorobotic exploration. In *Proceedings of the Congress on Evolutionary Computation at the 2002 IEEE World Congress on Computational Intelligence*, May 2002.

[137] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160, San Diego, CA, 2001. ACM, ACM Press.

[138] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2002.

[139] Madhavi W. Subbarao. Dynamic power-conscious routing for MANET's: an initial approach. *Journal of Research of the National Institute of Standards and Technology*, 1999.

[140] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 832–839, 1997.

[141] D. Sun and H. Man. Performance comparison of transport control protocols over mobile ad hoc networks. In *The 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC)*, September 2001.

[142] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[143] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[144] H. Tagaki and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, pages 246–257, 1984.

[145] Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 175–186. ACM, ACM Press, 2003.

[146] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5:97–116, 1999.

[147] Chai-Keong Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communications*, pages 1–36, March 1997.

[148] traceroute. Available online at `ftp://ftp.ee.lbl.gov/traceroute.tar.gz`.

[149] A. Tsirigos and Z. J. Haas. Multipath routing in the presence of frequent topological changes. *IEEE Communications Magazine*, 39, November 2001.

[150] Robbert van Renesse. The importance of aggregation. In André Schiper, Alex A. Shvartsman, Hakim Weatherspoon, and Ben Y. Zhao, editors, *Future Directions in Distributed Computing*, number 2584 in Lecture Notes in Computer Science, pages 87–92. Springer, 2003.

[151] Robbert Van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[152] Robbert van Renesse, Yaron Minsky, and Mark Hayden. A gossip-style failure detection service. In Nigel Davies, Kerry Raymond, and Jochen Seitz, editors, *Middleware '98*, pages 55–70. Springer, 1998.

[153] Z. Wang and J. Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *ACM Computer Communication Review*, 22(2), 1992.

[154] Duncan Watts. *Small Worlds: The Dynamics of Networks Between Order and Randomness*. Princeton University Press, 1999.

[155] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[156] R. Wehner and B. Lanfranconi. What do the ants know about the rotation of the sky? *Nature*, 293:731–733, 1981.

[157] WHO: Protection of the Human Environment. Water and Sanitation: Water supply and sanitation sector monitoring. Available online at `http://www.who.int/water_sanitation_health/wss/Monitoring2.html`, January 2003.

[158] J. Wieselthier, G. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of the IEEE INFOCOM 2000 Conference*, pages 585–594, 2000.

[159] Otto Wittner and Bjarne E. Helvik. Cross-Entropy Guided Ant-like Agents Finding Cyclic Paths in Scarcely Meshed Networks. In *The Third International Workshop on Ant Algorithms, ANTS'2002*, Brussels, Belgium, Sept 2002.

[160] Otto Wittner and Bjarne E. Helvik. Cross Entropy Guided Ant-like Agents Finding Dependable Primary/Backup Path Patterns in Networks. In *Proceedings of Congress on Evolutionary Computation (CEC2002)*, Honolulu, Hawaii, May 12-17th 2002. IEEE.

[161] G. K. Zipf. *Psycho-Biology of Languages*. Houghton-Mifflin, 1935.