Karlsruhe Institute of Technology (KIT)
Institute of Telematics

TELEMATICS TECHNICAL REPORTS

On the robustness of the BlåtAnt-S protocol under churn

Amos Brocco
Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany
brocco@kit.edu

August, 3rd 2011

TM-2011-5

ISSN 1613-849X

http://doc.tm.uka.de/tr/

Institute of Telematics
Karlsruhe Institute of Technology (KIT)
Zirkel 2, 76131 Karlsruhe, Germany

On the robustness of the BLÅTANT-S protocol under churn

Amos Brocco
Telematics Institute
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Email: brocco@tm.uka.de

***Abstract*—BLÅTANT is a peer-to-peer overlay management protocol which aims at maintaining an optimized topology to reduce the cost of broadcasting a message to all peers. BLÅTANT is based on bio-inspired methodologies that employ ant-like mobile agents and pheromone trails. The overlay maintained by this protocol can be classified as self-structured, because the topology is maintained in an adaptive way in order to fulfill certain criteria, namely an upper bounded diameter and a lower bounded girth (length of the smallest cycle). In this paper we consider the -S variant of the algorithm and focus on its robustness and efficiency in various churn situations. Our goal is to provide an in-depth analysis of the behavior of the protocol in a simulated network using the OverSwarm simulation platform. The obtained results are compared with those of two traditional peer-to-peer protocols, namely CHORD and GIA, in order to achieve a better understanding of the benefits and drawbacks of the considered bio-inspired solution.**

I. INTRODUCTION

An essential step in the research and development of novel network protocols is the comprehensive study of their behavior under realistic conditions and accurate usage models. These details include bandwidth limits, latencies and jitter at the underlay level, as well as churn patterns at the higher level. In this paper we aim at conducting a detailed analysis of a variant of the BLÅTANT algorithm [1], a peer-to-peer overlay based on self-organization principles and bio-inspired techniques. The maintenance protocol optimizes the logical connections between peers in order to achieve a bounded diameter (i.e. the maximum distance between each pair of nodes) while preserving only a minimal necessary number of redundant paths for resiliency. Two versions of the algorithm have been developed, namely the BLÅTANT-R variant [2] and the BLÅTANT-S [3] one. A comparison of the underlying logic and the implementation details of both versions is available in [4]. In this paper we focus on the -S variant, as its working principles are simpler and better match the philosophy of bio-inspired solutions. The overlay maintenance protocol (which will be detailed in Section IV) employs different *species* of ant-inspired agents that perform simple tasks on the network, such as collecting information about visited nodes and creating new logical connections. As with many self-organized and bio-inspired solutions, the behavior of the protocol is non-deterministic, and is dependent on the behavior of each agent. Nonetheless, the collective behavior of the whole *colony* of agents converges toward a common goal of maintaining a connected and optimized topology. However, because some of the agents are partially driven by stochastic behaviors, and because of the overall complexity of the system (which is composed of several agents of different species), a satisfying analytical proof is difficult to achieve. Accordingly, the operation of the system can be easier to understand by means of simulations under controlled conditions. In [4] the performance of the algorithm was evaluated in a custom simulator which had limited support for realistic network conditions. While the obtained results were enough to determine the correctness of the proposed approach (in terms of optimization constrains and robustness), a comprehensive evaluation was not carried out and was left as future work. The present research aims at exploiting a more accurate simulation framework in order to determine the efficiency and robustness of the algorithm with increased detail. We deem that this step can provide valuable information and serve as a reference work to drive future developments of similar protocols based on bio-inspired techniques. To achieve our goal we employ a novel simulation platform called OverSwarm, which extends the well known OverSim peer-to-peer simulator by providing explicit support for ant-inspired protocols. By means of an extensive evaluation in diverse realistic situations that reproduce network phenomenons such as jitter, queuing effects and churn patterns, we aim at providing a detailed report on the behavior of BLÅTANT-S. Of particular interests is the robustness of the overlay when faced with churn patterns that replicate common user behaviors in peer-to-peer networks, such as the ones based on lifetime models following Pareto or Weibull distributions. Another concern is the network traffic generate by mobile agents, and how it is affected by churn. Furthermore, we aim at comparing BLÅTANT-S with two traditional approaches (namely CHORD [5] and GIA [6]) under the same conditions, in order to determine the benefits and drawbacks of the considered bio-inspired solution. The remainder of this paper is organized as follows: in Section II we list the research goals and the research questions that are going to be answered by the present study; in Section III we detail the OverSwarm simulation platform, and discuss its architecture. In Section IV we detail the BLÅTANT-S algorithm and present some excerpts of its implementation using OverSwarm. In Section V we briefly discuss two traditional protocols that are used in

our comparative evaluation, whereas in Section VI we present the considered evaluation scenarios. Section VII presents and discusses the obtained results, whereas Section VIII provides our conclusions on this work and some insight on future research.

II. Research goals

The present work aims at improving our understanding of the BLÅTANT-S algorithm by providing additional simulation data. Although the protocol has already been extensively evaluated under different network and usage conditions in [4], important questions remain open. Accordingly, we set the goals of this research as follows:

- Analyze the behavior of BLÅTANT-S in realistic network conditions using a discrete time simulator that replicates latency, queuing effects, and jitter.
- Validate the robustness and reliability results presented in [4] by performing extensive evaluation the bio-inspired protocol under churn.
- Evaluate the use of the OverSwarm platform to support the development of ant-inspired network algorithms.
- Compare the reliability and robustness of bio-inspired solutions such as BLÅTANT-S with traditional approaches, namely GIA [6] and CHORD [5].
- Compare the bandwidth requirements of bio-inspired and traditional solutions.
- Evaluate the robustness as well as the adaptiveness of BLÅTANT-S versus traditional approaches in the event of network failures.

To achieve these goals we employ a different simulation platform than previous experiments, and thus a complete rewriting of the protocol implementation is necessary.

III. Simulation platform

To conduct our evaluation we implemented the BLÅTANT-S algorithm on the OverSwarm platform [7]. The main benefits of our choice are the availability of a realistic underlay model, as well as churn generators that enable accurate testing scenarios. In this section we briefly describe the architecture of the OverSwarm platform and highlight its main features.

OverSwarm is an extension of the popular OverSim [8] simulation platform, which is based on the discrete event simulator OMNET++ [9]. The framework is modular, and each component communicates by means of messages sent through communication channels linking input and output ports. Whereas OverSim focus is put on traditional peer-to-peer protocols that have nodes exchanging messages, OverSwarm concentrates on mobile agent ones. The mobile agent paradigm involves *intelligent* entities travelling across the network and performing some tasks on visited nodes. With strong migration capabilities, each agent carries its own execution status; in the case of transparent migration this status is automatically saved and restored when the agent moves from one node to another. This important difference between traditional and agent protocols is reflected in the development model of OverSim and OverSwarm. With the former the protocol is defined by the information carried by messages exchanged by peers, and by the response behaviors of a peer upon receiving and parsing a message. In OverSwarm the implementation of a bio-inspired algorithm is decoupled between the agent behavior (which also defines the information sent over the network) and the behavior of each peer, which can be either periodical or triggered by the reception of incoming data. OverSwarm enforces this decoupling by dividing the implementation of a protocol between one ore more C++ modules, which define the functionalities and information stored by each peer, and the behavior of agents, which is defined using a Lisp-like language. Agents can access data managed by the node through specially defined methods that can be invoked from Lisp. The use of a specific language for the definition of agent's behavior also enables strong and transparent migration capabilities, which is typical of ant-inspired protocols. Further details on the OverSwarm framework can be found in [7].

IV. Algorithm description

The BLÅTANT family of distributed algorithms maintains a peer-to-peer overlay with a continuously optimized topology that should reduce the communication overhead generated by flooding protocols. In this context, two algorithms have been developed: BLÅTANT-R and BLÅTANT-S. The former implements a more precise optimization process that consumes more traffic and imposes a slightly higher computational complexity on each node, while the latter follows a simpler approach that still provides satisfactory results, as proven in [4]. These algorithms are based on bio-inspired principles that mimic the behavior of insects; in particular, the swarm intelligence [10] and the ant colony optimization [11] paradigms are employed. In this regard, the network can be viewed as the environment where insect-like software agents live and migrate: each peer represents a nest were incoming agents can access local service and new agents could be generated. In contrast to traditional network algorithms, the protocol is not only defined by the operations executed by nodes and the exchanged information, but also from the behavior of each agent. In the following the details of the algorithm will be presented, and the necessary links between the formal description and the actual implementation using OverSwarm will be provided to illustrate the benefits of our approach.

A. Optimization process

BLÅTANT-S does not only strive to maintain a connected overlay, but also aims at optimizing the topology of logical connections between peers in order to reduce the global communication cost when multicasting information across nodes (for example using flooding [12], [13], [14], gossiping [15], or random walk [16], [17], [18] protocols). The overlay optimization process is controlled by two simple rules that determine the creation of new logical links and the destruction of existing ones. The goal of the link creation phase is to limit the diameter of the network according to an optimization parameter D. More specifically, a new link is created between

two peers if a distance greater than

$$2D - 1$$

is observed. Conversely, removing links aims at breaking up cycles in the topology that might cause message retransmission when the overlay is flooded with queries. The rule for breaking up cycles is used to determine if two neighbor peers must be disconnected, and applies when there exist an alternative path that connects them whose length is less than

$$2D - 3$$

hops. To determine the distance between peers as well as to perform connection and disconnection, ant-like agents are employed.

B. Pheromone trails

Pheromone trails are used as a form of indirect communication between insects. Individuals of a colony can leave some chemical in the environment to mark interesting paths, which can be later exploited by other agents. Chemicals naturally evaporate with time, rendering trails less desirable: to avoid so, concentrations can be periodically reinforced to keep them alive. Artificial ants replicate pheromones concentrations by associating a numerical value to links in the overlay that are stored in a hashtable on each node: evaporation is simulated by periodically reducing this value.

BLÅTANT-S employs pheromone trails in order to detect node failures and abrupt disconnections in high-churn situations. Each agent deposits pheromone on each path while migrating; the concentration of the pheromone determines the liveness of nodes and enables the system to react accordingly. Two types of pheronomne trails, associated with each neighbor on each node, are employed: beta (β) and gamma (γ). On one hand, beta trails are reinforced by ants arriving on a node, and are used to track communication originating at neighboring nodes and detect disconnections; if a beta trail completely evaporates a peer can deduce that the corresponding neighbor failed (i.e. abruptly disconnected) or that some communication problem exists, and initiate a recovery procedure to connect with that part of the overlay, namely by contacting the last known neighbors of the peer that leftq. On the other hand, gamma pheromone trails are used for two purposes: first, they ensure that all paths in the overlay are equally explored by Discovery agents, second they trigger ping traffic on the corresponding path to signal aliveness of the node should normal traffic not be enough. To achieve the first purpose, Discovery agents (discussed in the following subsection) preferably follow paths with the lowest concentration trails; for the second purpose, the decay rate of γ trails is about two time faster than that of the β trails.

C. Agents

The communication protocol used by BlåtAnt is based on different species of ant-like mobile agents. Each agent has a different purpose and is able to transparently migrate from peer to peer to perform its tasks. In the following we present and discuss the behavior of the most significant species; for additional details on this subject please refer to [4].

```
(while 1 (begin
    (if (<= steps 0) (break))
    (if (inform vector) (clear vector))
    (push vector (getThisNode))
    (if (> (len vector) vectorlength) (erase vector 0))
    (var neighbors (getNeighbors))
    (remove neighbors previous)
    (foreach v in vector
        (remove neighbors v))
    (if (= (len neighbors) 0) (begin
            (set! neighbors (getNeighbors))
            (set! vector [])))
    (if (< (random) kappa)
        (set! nextStep (getLowestGammaTrail neighbors))
    else
        (set! nextStep (choose neighbors)))
    (if nextStep (begin
        (set! previous (getThisNode))
        (set! steps (- steps 1))
        (if (not (migration nextStep)) (end))))))
```

Fig. 1. Discovery Agent

1) Discovery: Discovery agents are periodically created on each peer, according to some *respawn* probability. Each agent wanders on the overlay for a predefined maximum number of steps (hops in the overlay): on each peer the identifier of the node (typically its IP address and port) is stored into a bounded size vector. Subsequently, a candidate target node is chosen among the neighbors of the current peer, such that previously visited peers are not considered. If no candidate is available, all neighbors are reconsidered. The ant can choose between exploration of the overlay, by choosing a random neighbor, or exploitation, by choosing its target according to the concentration of local γ trails. More specifically, with a given probability *k*, the agent migrates to a random neighbor, otherwise it choses the neighbor associated with the lowest concentration of gamma pheromone.

The behavior of the agent has been easily implemented with OverSwarm, as shown in Figure 1. The agent executes a series of operations on the current node, before migrating to the next peer. The `steps` variable keeps track of the current number of hops traveled in the overlay: when all possible steps are expired the ant is killed. The addresses of visited nodes are stored in a dynamic list `vector`. Each time a new address is added to the list, its size bounds are check, and exceeding information is eventually dropped. Selection of the neighbor with the lowest gamma pheromone concentration is implemented as a separate function named `getLowestGammaTrail`. The `migration` function not only performs the actual migration but also reinforces both the γ pheromone trail at the source node, and the β one upon arrival on the target node.

2) Optimization Link: Optimization Link agents are used to create new logical links between nodes in order to optimize the topology of the overlay. Agent are deployed by peers that want to connect to some other node because the connection rule applies; the ant starts from the node requesting the connection, and migrates to its target (Figure 2). If the conditions on the hop distance between the two nodes is fulfilled (as determined

```
(virtual $target)
(var source (getThisNode))
(if (migrate $target) (begin
    (var d (getEstimatedDistance source))
    (if (and (> d 0) (< d (- (* 2 (D)) 1 ))) (end))
    (var isAcceptedConnection (connect source))
    (if isAcceptedConnection
        (if (migrate source)
         (connect $target)))))
```

Fig. 2. Optimization Link Agent

by local cached information on the target node) the logical link is created and the agent migrates back to the source peer to complete its operations. The `getEstimatedDistance` function returns the estimated number of hops separating the current node from the source (according to locally cached information), whereas the `connect` function creates a new logical link to the specified node and initializes the corresponding pheromone trails.

3) Unlink: Unlink agents are used to remove logical links between nodes when the disconnection rule applies or when a node wants to leave from the overlay. The Unlink agent migrates between the nodes to remove the logical link information and clear the associated pheromone trails on both.

4) Construction Link: Construction Link ones are used to perform an initial connection between two peers in order to let a new peer join the overlay. These species of ant are also used to recover connectivity in the event of a failure. Accordingly, the logical links created by these agents have no optimization purposes (i.e. the connection rule must not necessarily be satisfied).

Periodically, using the available information gathered from other nodes through *Discovery* agents and *Update Neighbors* agents, nodes detect and break up additional small cycles of lengths of 3 and 4 hops.

TABLE I
BLÅTANT-S PROTOCOL PARAMETERS

Parameter	Value	Description
β decay	60s	Evaporation time for the β pheromone
γ decay	30s	Evaporation time for the γ pheromone
α max size	28	Maximum number of entries in the α table
vector max size	15	Maximum number of entries in Discovery ant vector
D	5	Topology optimization parameter
Max. optimization links	6	Maximum optimization links on each node
Max. node degree	8	Maximum outgoing links on each node
Reconnection interval	30s	Time to wait between connection requests
α max age	1800s	Time before discarding old information
Respawn probability	0.05	Respawn probability for Discovery ants
Respawn interval	100s	Respawn interval for Discovery ants and Update Neighbors ants
Recovery delay	15s	Interval between recovery retries when node is isolated

D. Protocol parameters

In all the experiments presented in the following of this paper the configuration parameters of BLÅTANT-S have been kept constant. A detailed overview of the considered values is presented in Table I. These default values are based on our previous experience with the protocol and provide a suitable configuration for the considered evaluation scenarios.

V. COMPARISON WITH TRADITIONAL APPROACHES

As highlighted in the introduction, bio-inspired systems might represent an interesting and valuable alternative to traditional protocols. In order to foster the adoption of such methodologies in the realm of distributed applications we argue that a systematic comparison of both approaches (traditional and bio-inspired) is necessary. For this purpose the evaluation presented in this paper also considers two existing P2P protocols, namely CHORD [5] and GIA [6]. Our analysis will focus on both the robustness of these systems in dynamic conditions, as well as the network overhead resulting from the topology maintenance tasks. In the following we briefly discuss the inner workings of both CHORD and GIA. We should stress that only the topology maintenance part of both GIA and CHORD are considered for comparison with BLÅTANT-S; more specifically, we are interested in the ability of the management protocol of maintaining a connected overlay, on the characteristics of the resulting topology, and on the amount of maintenance traffic.

A. Chord

CHORD [5] is a structured overlay implementing a distributed hashtable storage solution. Each node is assigned with a unique identifier of m bits within a circular space of size 2^m. The topology is organized as a logical ring where nodes are ordered according to their identifier. Each node knows its successor in the ring, i.e. the node whose identifier follows in the identifiers' space. When employed as a DHT, content shared by nodes is assigned an key (typically a hash of the data) of m bits, and is published on the node referred to as $successor(K)$ so that the identifier of the latter matches K or follows it. In order to find $successor(K)$ in the overlay, either to build up the finger table or to lookup for a key, a node starts by querying known nodes starting from the one that appears closer to K, and repeats the process until the target peer has been found. To speed up the look-up operation, each node n maintains a routing table (called *finger table*) of size m that contains the identifiers of other peers in the ring: the entry at the i^{th} position ($1 \leq i \leq m$) in the finger table corresponds to the first node that succeeds n by at least 2^{i-1} hops in the ring, i.e. $successor(s)$ with $s = n + 2^{i-1}$. The routing cost in a CHORD overlay of N peers is of $O(logN)$ hops. The finger table is updated by performing queries. In the context of our evaluation, both predecessor, successors, and finger nodes are considered as neighbors. A node joins the overlay by sending a connection request to one of the existing peers. The position of the newly inserted node in the ring is determined by its identifier: because this operation

involves querying the network, the cost of a node join is $O(log^2 N)$ messages. When a node joins the overlay or leaves the system, the successors pointer and finger tables of nearby nodes in the ring are updated by means of a *stabilization* procedure. To improve resilience in the event a successor abruptly disconnects from the overlay, each node maintains a list of nodes that succeed it in the ring. Peers periodically ping their predecessors, successors and fingers, in order to check for failed nodes. If a predecessor fails the node repeats the joining process, conversely if a successor or a finger fails a substitute is chosen among the known backup addresses. For our evaluation we only consider the topological properties of CHORD. Accordingly we do not perform any resource discovery tasks on the overlay, nor we fully exploit the key look-up mechanisms implemented by this peer-to-peer system, apart for letting a new node join the overlay by querying its position in the ring. The protocol parameters as used in our simulations are listed in Table II.

TABLE II
CHORD PROTOCOL PARAMETERS

Parameter	Value	Description
m	160bits	Size of the node's identifier
joinRetry	2	Retries to join the overlay
joinDelay	10s	Minimum wait between successive join retries
stabilizeRetry	1	Retries after failed stabilize
stabilizeDelay	20s	Minimum wait between successive stabilizations
fixfingerDelay	120s	Interval between finger table fix process
checkPredecessorDelay	5s	Interval for checking predecessor
successorListSize	8	Maximum number of peers in the successor list

B. Gia

GIA [6] is a peer-to-peer protocol for unstructured overlays that strives to address the scalability problems of its ancestor, GNUTELLA [19]. The relationship between the two protocols is reflected in the basic messages that both employ to discover new peers in the network, namely ping and pong messages. However, GIA incorporates a self-organizing topology management process that improves search operations, and makes use of different querying techniques (like random walks) rather than just relying on flooding with limited TTL (Time-To-Live). Nodes rearrange their neighbors in order to maintain links with peers that are well connected and are able to sustain the resulting traffic; in this regard, each peer try to make high-capacity nodes (nodes with high bandwidth and storage) easily reachable (within few hops), which results in the latter receiving the largest portion of the requests. To evaluate the quality of current connections during the topology adaptation process, each peer determines its *satisfaction level* (between 0 and 1). This value is related to the node's capacity and that of its neighbors, as well as to its degree. Each peer maintains a cache of addresses of other nodes (which are candidate neighbors); the cache is populated either from a list of well-known nodes, or through ping-pong messages. Should the level of satisfaction be too low, new neighbors are randomly chosen from the cache. In order to control the maximum traffic each peer receives, an active flow control mechanism is implemented. Each node accepts incoming queries based on its capacity: each peer periodically sends a token to its neighbors, and the latter can only deliver a query back to the former as long as a token has been received. If a neighbor does not make use of its token, it is marked as inactive, and the spared bandwidth is redistributed among its siblings. Queries traverse the overlay typically using a biased random walk scheme, in order to be directed toward high-capacity nodes that have a higher probability of fulfilling the request. The parameters of the protocol employed in our experiments are detailed in Table III: the chosen values are based on the default configuration provided by OverSim; the maximum number of neighbors is based on a experiments presented in [20], which employed overlays of comparable size as in our simulation scenarios. The capacity of each node is determined uniformly at random at node's initialization.

TABLE III
GIA PROTOCOL PARAMETERS

Parameter	Value	Description
maxNeighbors	20	Maximum number of neighbors
minNeighbors	3	Minimum number of neighbors
topAdaptationInterval	120s	Interval for re-evaluating node's satisfaction level and adapt topology
updateDelay	60s	Interval for updating neighbors about current node's capacity
maxHopCount	10	Maximum TTL for sent messages
messageTimeout	180s	Message timeout
neighborTimeout	250s	Neighbor timeout
sendTokenTimeout	5s	Interval for sending tokens to neighbors
tokenWaitTime	5s	How much to wait for a new token
keyListDelay	100s	Delay when sending a new key to neighbors

VI. EVALUATION SCENARIOS

We evaluate the behavior of BLÅTANT-S under both different churn-models and in communication failure situations. In particular we consider various probabilistic lifetime distributions, such as Pareto and Weibull, as well as random node removal with regular periodicity. For this purpose, the dedicated support of the underlying OverSim platform is fully exploited: the employed churn models are already implemented and can be easily included in simulations. In contrast to previous evaluations, such as the ones presented in [4] and [2], the aim of this research is to establish the robustness of the protocol using more realistic usage models, as previous results were based on simplistic churn patterns, with nodes periodically added and removed to the network following a Poisson process. Moreover, we also conduct experiments to establish the communication fault tolerance, by simulating packet loss. Each experiment reproduces 6 hours of network activity, where we distinguish between an initialization phase and the *real simulation*: in the former, nodes just join the overlay with a constant rate and no node quits the network;

conversely, in the latter the joining and leaving of peers is determined by the considered churn model. It should be noted that no intermediate *transition* phase is employed, although supported by OverSim.

A. Traffic simulation and analysis

To determine the generated network overhead, the average bandwidth consumed by each node is measured. Concerning BLÅTANT-S, the actual message size resulting from OverSwarm's serialized agent state is considered. Although more efficient implementations could potentially reduce the amount of information exchanged by nodes, we also aim at proving the effectiveness of OverSwarm for the deployment of real systems using the same core components. The reported bandwidth consumption only considers the payload size, not the complete packet size (which include UDP headers). The SimpleUnderlay model implemented by OverSim was chosen: despite the name, latency, bandwidth, and jitter are fully reproduced by the simulator. Nodes are mapped into a 2-dimensional Euclidean space; latency is determined by means of the distance between peers in this space, and is based on Internet latency measurements obtained in the framework of the CAIDA/Skitter [21] project so that node positions are set to produce realistic latencies between all peers in the simulation. The employed sample includes a pool of 15000 candidate peers.

B. Churn models

In the initialization phase of all experiments an expanding overlays is simulated: all nodes are connected to the network with a frequency of 1 node every second. Nodes rely on a globally accessible bootstrapping service to learn about existing overlay nodes to connect to. Subsequently, to simulate the dynamics of users joining and leaving the network, we employ some of the churn models implemented by OverSim, that reproduce typical patterns observed in real systems. Amongst the available models, we considered two lifetime based patterns (Weibull and Pareto), as well as simpler uniform random pattern. On one hand, Weibull churn reproduces an average lifetime (expected value) of 10000 seconds, and a distribution parameter k equal to 1. On the other hand, Pareto churn is based on [22], and uses an average lifetime of 10000 seconds and an α parameter equal to 3. The minimal lifetime x_m resulting from such parameters is thus 6666 seconds. The corresponding cumulative distribution functions of the lifetime for both models is plotted in Figure 3. These distributions are also used to trigger the creation of a new node that will join the overlay: new nodes are created after a *dead time* drawn from the same probability function and an average of 10000 seconds. The Random churn model is based on the following logic: every 4 seconds a node a random number is drawn, and with 50% probability an existing node is removed from the network, or a new one is added. In all experiments BLÅTANT-S disconnection is performed in an abrupt manner, to better evaluate the recovery mechanism of the algorithm. Based on the obtained results, the Weibull churn model results

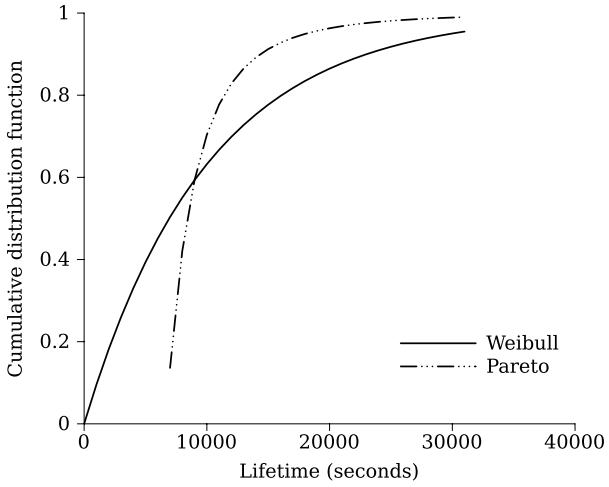

Fig. 3. Lifetime cumulative distribution function

in an average of 32 nodes added and 32 removed every 500 seconds in an overlay with an average of 512 nodes, and 65 nodes added and removed with an average of 1024 nodes. Conversely, Pareto churn adds and removes an average of 34 nodes in a 512 nodes overlay every 500 seconds, while adding and removing 65 in a 1024 nodes network. Finally, the random churn pattern inserts and deletes 68 nodes in the 512 nodes scenarios in the same period of time, and 75 nodes in overlays of doubled size. From these figures, it is clear that Pareto and Weibull churn models produce a slightly lower dynamicity in the network compared to the simple random model with 1024 nodes, and a considerably lower dynamicity with 512 nodes. As a baseline for comparison, simulations with stable overlays without churn are also carried out.

C. Fault tolerance

Whereas churn models enable an evaluation of the resilience of the overlay when nodes join and leave, separate scenarios are required to determine the fault tolerance capabilities of each system. In this regard, we simulate the loss of transmitted packets and determine the effects on both the resulting traffic as well as on the topology (which provides a hint of the robustness of the system). All the aforementioned experiments where thus repeated by introducing a probability of losing packets during transmission; more specifically, each time a UDP packet is transmitted it has a 10% probability of being lost. Accordingly, packet loss influences the ratio between sent traffic and received traffic. Because transmitted protocol messages are relatively simple and their size is within the maximum length of a UDP packet payload, we assume that our packet loss chance leads to a message loss of equal probability. To account for the failed delivery of such packets in our bandwidth measurements, traffic results will only focus on the amount of data sent by nodes.

VII. RESULTS

In this section we report the results obtained in our experiments in relation with the goals set in Section II. The

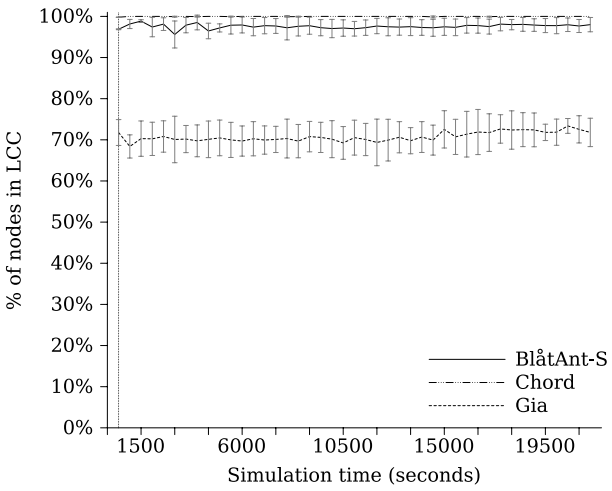

Fig. 4. Weibull, 512 Nodes

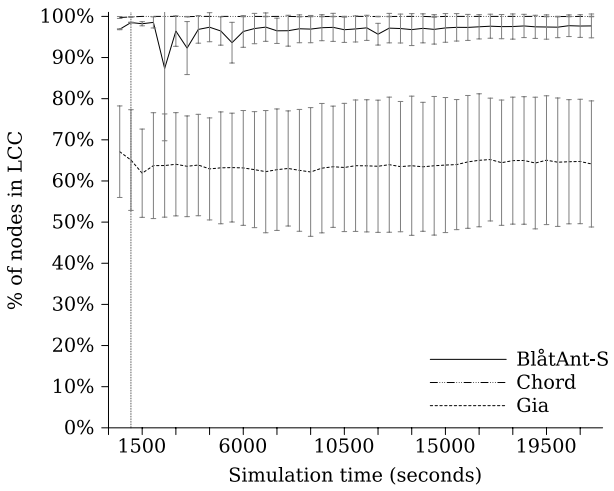

Fig. 5. Weibull, 1024 Nodes

presented data refers to an average over 3 simulation runs, with the standard deviation clearly indicated by means of vertical error bars. The dotted vertical line denotes the end of the initialization phase, which occurs at about 512 seconds into simulation in overlays of 512 nodes, and at 1024 seconds in overlays of 1024 nodes, in compliance with a join rate of 1 node every second. Our evaluation is conducted along three main axis, namely connectivity, topology measurements, and bandwidth consumption. For each concern we detail and discuss here only the most significant results and present only the corresponding graphs: additional graphs are available in Appendix A.

A. Connectivity

Connectivity experiments aim at evaluating the ability of the management protocol to maintain an overlay graph that consists of a single, large partition. This property is important when multicast communication is to be carried out on the overlay, as it ensures that all nodes can be potentially reached by simply flooding the network. In this respect, we report and compare the size of the largest connected component (LCC) as measured in our scenarios with different churn models, overlay sizes, and communication reliability.

1) Connectivity with reliable communication: The first set of results concerns the connectivity of the overlay when communication between peers is reliable. More specifically, we do not consider packet loss, and communication is only affected by latency, jitter, and queuing effects.

Figures 4 and 5 depict the evolution of the LCC in overlays of 512 and 1024 nodes respectively, using the Weibull churn pattern. At both scales, the deterministic behavior of CHORD enables the algorithm to maintain a a single connected component that spans over all peers, whereas BLÅTANT-S attains an average of 98% in the 512 nodes overlay, and 97% in the 1024 nodes overlay. This result is unsurprising, as the connection and recovery procedures of BLÅTANT-S are non-deterministic and based on *best-effort* premises. The process to connect a new peer to the overlay is conducted in multiple steps, as the algorithm tries to join the newly created node to an existing peer which has a small degree: this might require the agent responsible for the initial connection to be forwarded for several hops in the overlay, cause delays in the joining process, and explain the perceived inability to maintain a LCC of 100%. It shoud be noted that the LCC remains nonetheless close to this value, and the size of the largest partition remains almost constant during the simulation. This results practically rules out issues with the recovery procedure, and provides a an insight of the robustness of BLÅTANT-S, whose performance can be considered as remarkable when compared to CHORD. Both protocol manage to cope with churn, although BLÅTANT-S nodes achieve that using only limited information for recovery, namely the neighbors of the node that left the network, whereas CHORD peers can typically rely on their predecessor, list of successors, and finger table.

Surprisingly, GIA is not able to maintain connections between all nodes in the overlay, despite showing the highest number of logical links in the overlay (as it will be detailed in the following subsection). The average size of the largest connected component is of 72% in the 512 peers overlay, and 65% in the 1024 peers overlay. Moreover, the behavior of GIA is highly unstable as the overlay is scaled up to 1024 nodes. This phenomenon could be explained by the topology adaptation mechanism that favors neighbors with the highest degree: if one of such neighbors disconnects the overlay is greatly affected and fails to properly recover.

With the Pareto churn pattern (Figures 6 and 7) BLÅTANT-S performs slightly better in the 512 nodes scenario, being able to maintain a LCC of 99% of the nodes. It is however possible to notice a small performance drop in the 1024 nodes overlay, down to 95%. These numbers remain nonetheless stable during the simulation, meaning that they are again due to the joining delay rather than to a failure of the recovery mechanism. CHORD displays once more its ability to ensure a connected overlay, confirming that the protocol can cope well

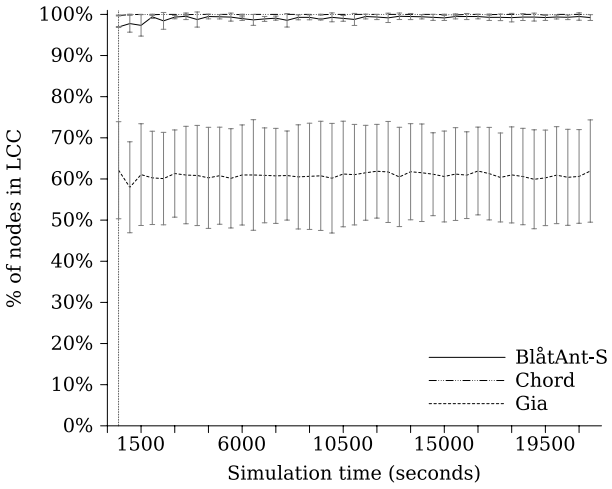

Fig. 6. Pareto, 512 Nodes

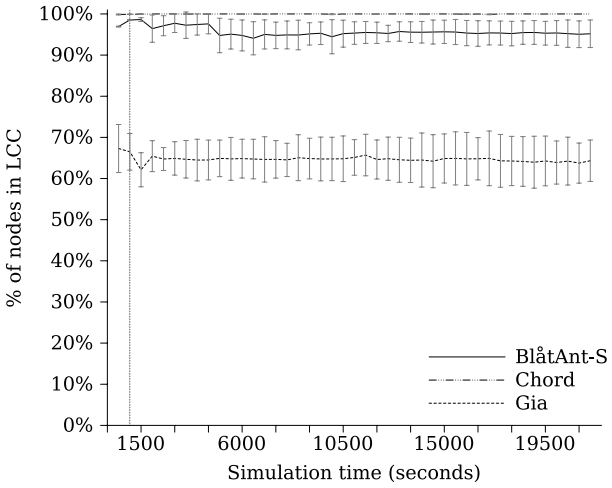

Fig. 7. Pareto, 1024 Nodes

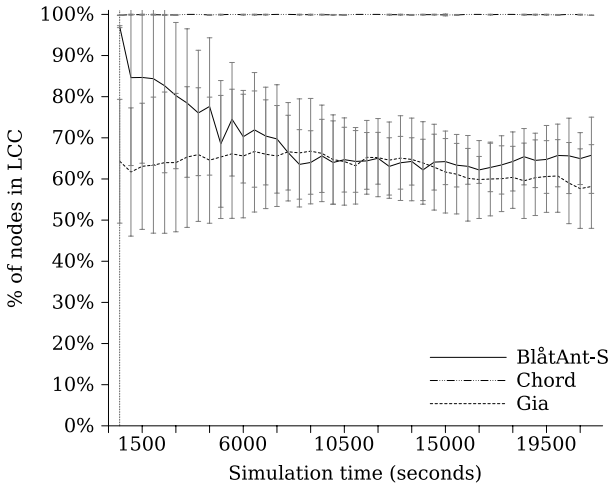

Fig. 8. Random, 512 Nodes

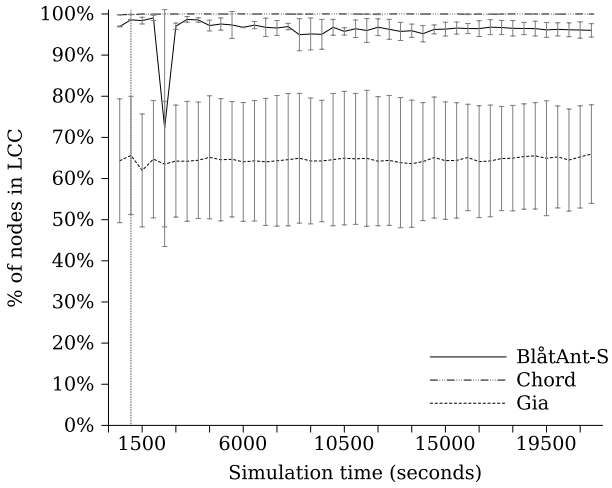

Fig. 9. Random, 1024 Nodes

with the considered churn model.

As with previous experiments, GIA fails to maintain all nodes within the largest connected component. As shown by the results, the behavior of the protocol when faced with the Pareto churn model is similar to that of the Weibull one: only about 60% to 65% of the nodes are in the largest partition in both the 512 and the 1024 peers overlay. It should be noted that this does not necessarily affect the performance of the querying mechanism implemented by GIA, as the use of one-hop replication greatly reduces the risk of misses, even though not all nodes can be reached by the query. However, for group communication purposes GIA might not be well suited.

When focusing on the churn pattern that results in the highest dynamicity, namely Random, it is possible to notice how a smaller overlay penalizes both BLÅTANT-S and (in a less important way) GIA. The faster rate of nodes joining and leaving the overlay greatly affects BLÅTANT-S, the performance of which drops to 65% before initiating a slightly recover. This result can be attributed to the low number of links that need to be created in order to optimize its topology by bounding the diameter. As the overlay is small, very few additional connections are required, thus the robustness of the topology is inherently affected.

By observing the results obtained in the overlay of 1024 nodes, an important drop can be noticed at the beginning of the simulation. By analyzing the error bars we can nonetheless classify such behavior as transient. Overall, an increased number of peers restores the performance of BLÅTANT-S to the same level observed with other churn models. This leads us to affirm that the robustness of our bio-inspired protocol also depends on the size of the overlay: the largest the overlay, the more robust the system becomes.

As a final result, we consider the connectivity in a stable overlay (Figures 10 and 11): unsurprisingly, both CHORD and BLÅTANT-S are able to maintain full connectivity between all nodes. Reflecting the previously presented results, GIA still fails and provides an average LCC of 65% of the nodes. Despite a stable overlay, the variance in the results obtained with

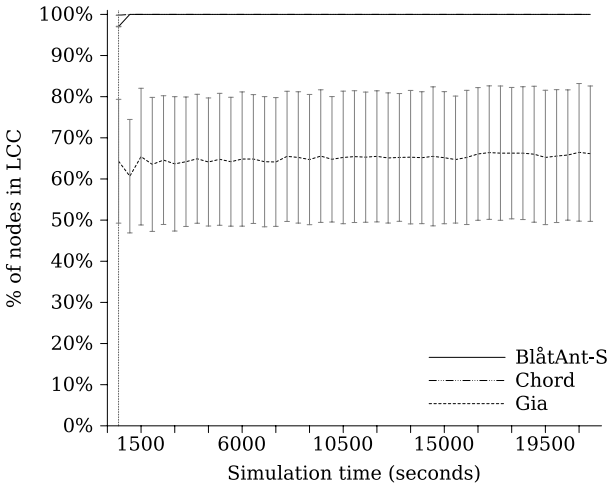

Fig. 10. No churn, 512 Nodes

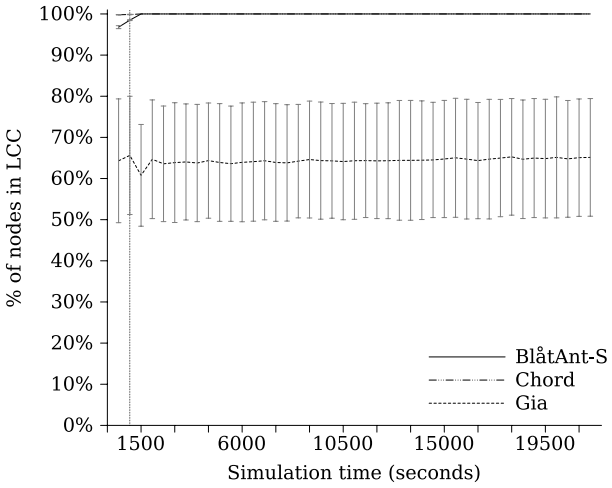

Fig. 11. No churn, 1024 Nodes

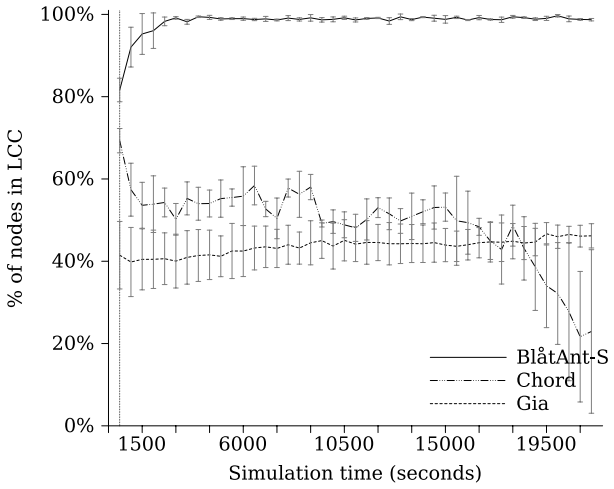

Fig. 12. Weibull, 512 Nodes, 10% drop

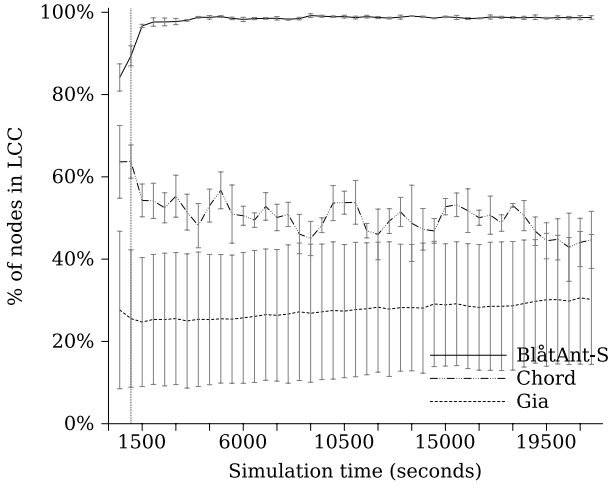

Fig. 13. Weibull, 1024 Nodes, 10% drop

the GIA protocol remains very high compared to BLÅTANT-S and CHORD, with the LCC spanning between 50% and 80%. With GIA it is thus very difficult to ensure that the majority of the nodes remain connected to the largest partition of the overlay.

2) Connectivity with unreliable communication: The second set of results concerning connectivity originates from experiments made with an unreliable underlying network, where each transmitted UDP packet has 10% probability of not reaching its destination. This scenario is more likely to happen when wireless communication is involved.

As shown in Figures 12 and 13, only BLÅTANT-S is able to cope with packet loss. In particular, the LCC increases to 99% with BLÅTANT-S, whereas it drops to around 20% with CHORD in a 512 nodes overlay and to 40% in overlays of 1024 nodes. On the other hand, GIA remains almost stable at about 40% in overlays of 512 nodes, and at 25-30% in overlays of 1024 nodes. As it will become clearer in the following of this paper, the robustness of BLÅTANT-S comes at a cost of an increased traffic and higher average node degree. Nonetheless, these results prove that BLÅTANT-S has a clear advantage over traditional protocols in the event of packet loss, and is able to adapt to such changes in the environment through emergent behaviors, as no specific response to packet loss is explicitly implemented. It should be noted that implementations of all three protocols using TCP communication could be used to overcome packet loss, at the expense of a higher network overhead.

Although not shown in the presented figures, the observed behavior of all three protocols with either no-chun or the Pareto churn model, as well as with either 512 or 1024 nodes, is similar to that observed with Weibull and 1024 nodes: BLÅTANT-S is able to maintain a LCC close to 100%, CHORD falls between 40% and 60%, whereas GIA remains around 25%-30%.

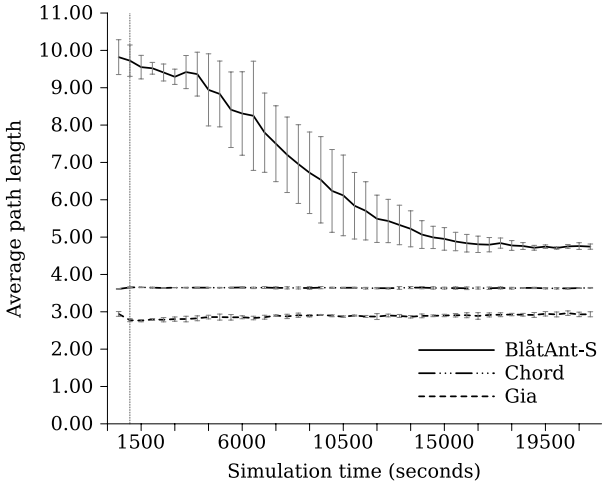

Fig. 14. Weibull, 512 Nodes

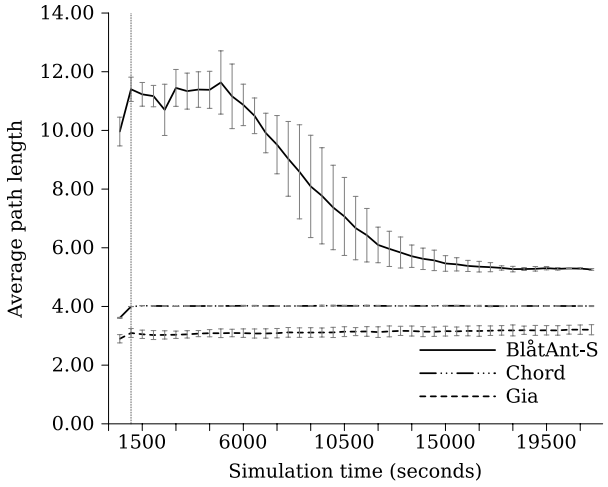

Fig. 15. Weibull, 1024 Nodes

B. Topology measurements

Apart from the size of the largest connected component we are interested in other topology measurements such as the average path length, the average node degree, and the length of the detected cycles. The first measure gives us an indication of the average number of hops a query needs to be forwarded in order to reach the majority of the nodes. The node degree determines the complexity of the overlay, and can be used to estimate the cost of flooding (as a message is forwarded through neighbors at each step). Finally, the length of cycles in the topology determines the likelihood of redundant message transmission, and thus relates to the efficiency of flooding protocols executed on the overlay. As for previous results, only the significant data is presented here, whereas additional results can be found in the appendix.

1) Topology measurements with reliable communication: The average path length in overlays of 512 and 1024 peers subject to the Weibull churn pattern is plotted in Figures 14 and 15 respectively. With both overlay sizes, CHORD and GIA exhibit the lowest path lengths, with 4 and 3 hops respectively, whereas BLÅTANT-S starts with a length close to 12 hops and subsequently reduces it to 5. This reduction can be explained both by the optimization process that takes place in the overlay, as well as with the additional connections that are created by recovery procedures triggered by churn.

By comparing the number of cycles of length less than 6 hops (Figures 16 and 17) the benefits of the optimization process of BLÅTANT-S are evident: whereas both CHORD and GIA topologies contain a large number of such cycles, BLÅTANT-S is able to minimize their number to about 200 in both the 512 as well as in the 1024 nodes overlay. We recall that for the evaluation of CHORD we consider as neighbors both the predecessor, the successor, the nodes in the successor list, and the nodes in the finger table. As the network scales up, CHORD doubles the number of cycles whereas GIA suffers from only a slight increase. Results obtained with both the Pareto and the Random churn patterns are very similar to the Weibull ones, and are reported in the appendix. A significant results is observed when no churn occurs in the network, where GIA generates about 1000 cycles more than in churn situations.

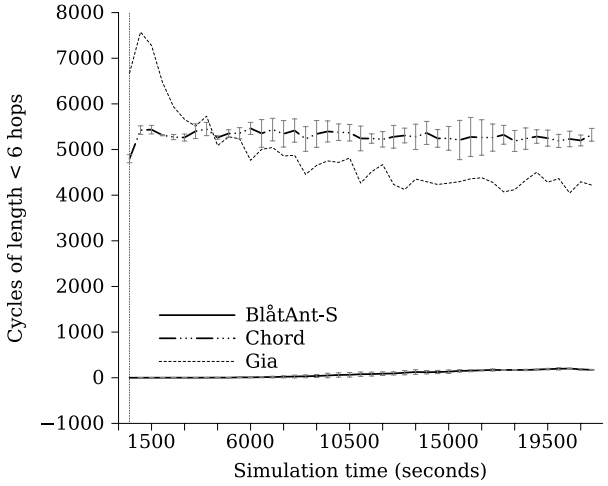

Fig. 16. Weibull, 512 Nodes

Finally, concerning the average node degree, in Figures 18 and 19 it is possible to notice that CHORD maintains the highest number of logical connections, with an average of about 10 in the 512 nodes overlay, and 11 in the 1024 nodes overlay. As with previous comparisons, we only present graphs for the Weibull case, as other churn models provide similar results. GIA maintains an average of 6 neighbors (out of a minimum of 3 and a maximum of 20), and does not show any significant variation when the scale of the overlay is doubled. BLÅTANT-S starts with an average of just 2 neighbors per node, but as the network is optimized and recovery from churn takes place, it increases this value to and average of 4 neighbors out of a maximum degree of 8.

2) Topology measurements with unreliable communication: When communication issues are introduced in the simulation two phenomenons can be observed: on one hand, BLÅTANT-

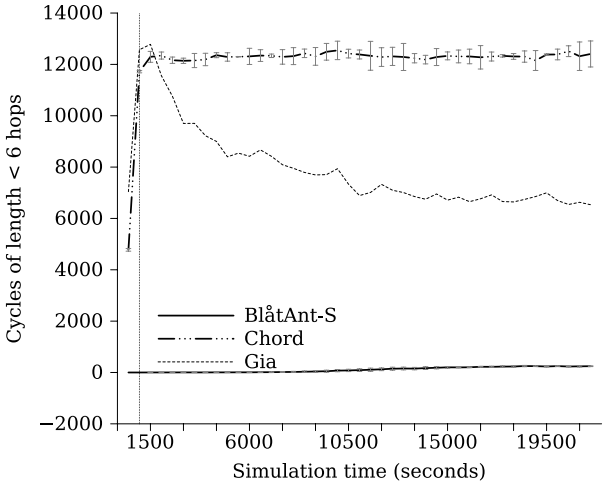

Fig. 17. Weibull, 1024 Nodes

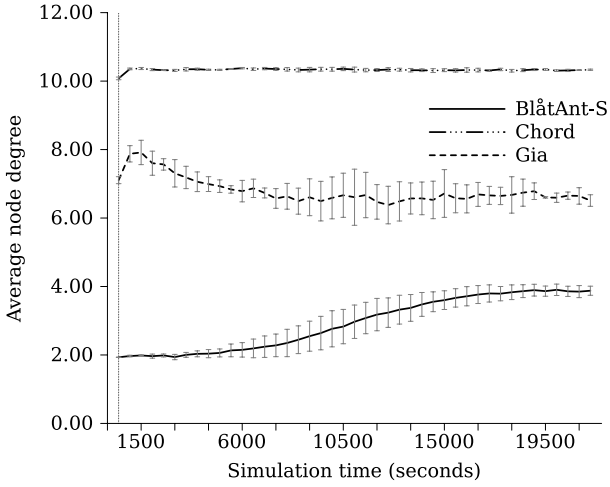

Fig. 18. Weibull, 512 Nodes

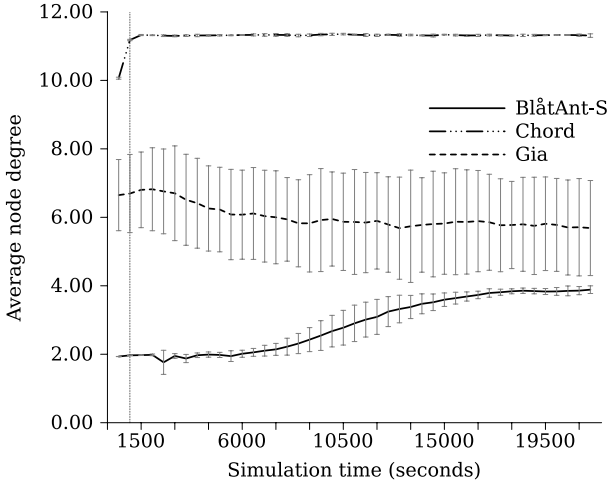

Fig. 19. Weibull, 1024 Nodes

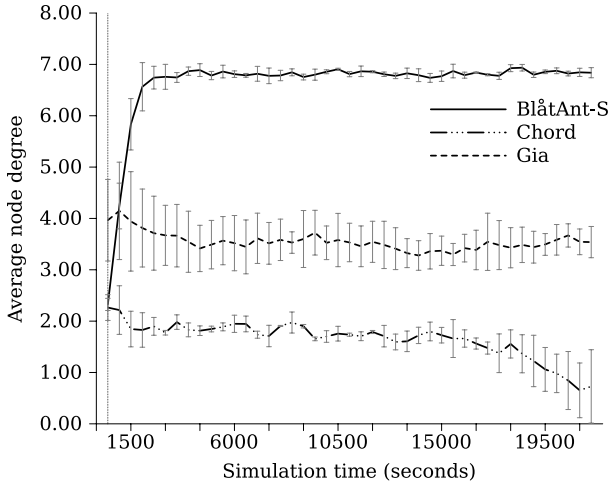

Fig. 20. Weibull, 512 Nodes, 10% drop

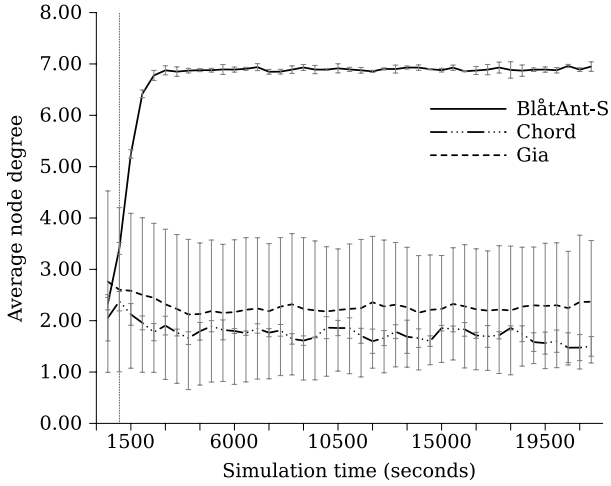

Fig. 21. Weibull, 1024 Nodes, 10% drop

S topologies become more complex in order to overcome the loss of information that could lead to a partitioning of the network; on the other hand, both CHORD and GIA wrongfully detect nodes' failure and drop existing connections causing a dangerous partitioning of the overlay. These phenomenons are clearly observed by looking at the average node degree with Weibull churn and 10% packet loss plotted in Figure 20 and 21: CHORD disconnects an average of 8 neighbors, whereas in GIA the node degree decreases by about 2 neighbors. In contrast, the average degree of BLÅTANT-S nodes increases to 7 in both the 512 and the 1024 nodes network, as a result of the recovery procedures that are started when pheromone trails completely evaporate. In BLÅTANT-S this behavior is completely natural, as recovery is carried out as if peers disconnect.

Concerning the average path length and the number of detected graph cycles it should be noted that, for CHORD and GIA, the result is biased by the fact that the topology becomes partitioned. Hence the obtained values are unreliable

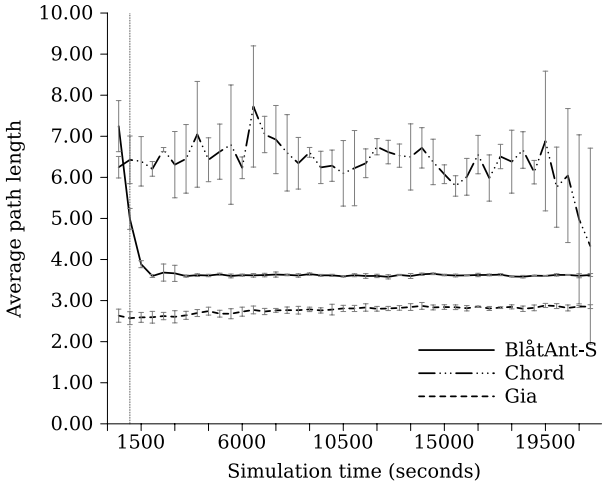

Fig. 22. Weibull, 512 Nodes, 10% packet drop

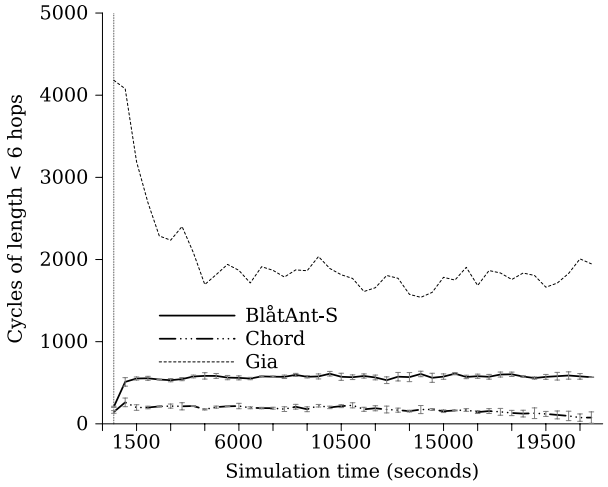

Fig. 23. Weibull, 512 Nodes, 10% packet drop

and do not reflect the real status of the network. As shown in Figures 22 and 24, the average path length under Weibull churn increases for CHORD up to 6 hops, but remains highly unstable. With GIA no significant variation can be observed, and the average distance remains around 3 hops. Conversely, for BLÅTANT-S the convergence toward a stable minimum (of less than 4 hops, compared to 5 in the reliable network scenario) is faster than in previous experiments. Results with different churn models and scale show similar results and a highly unstable behavior of the CHORD protocol.

Partitioning of the network affects the number of detected cycles in both GIA and CHORD: as the connectivity in both overlays fails, their topologies contain very few small cycles. Conversely, with packet loss BLÅTANT-S generates double the number of small cycles, due to recovery procedures that create new links that have not to match the optimization rule. This is generally not a problem, since we argue that the major concern when encountering communication failures is maintaining a connected overlay and ensuring that a large part of the peers are still reachable. Nonetheless, the number of cycles of length less than 6 is still minimal when compared with that of CHORD and GIA in normal conditions (reliable communication).

From the results presented in this section it is clear that BLÅTANT-S provides a clear advantage over GIA and CHORD when it comes to unreliable networks. The topology of BLÅTANT-S overlays stays connected and the optimization process still ensures that the diameter of the network remains within its bounds and that the number of small cycles is minimized. In this regard, the bio-inspired nature of the algorithm plays and important role, because the loss of some individual agents of the colony does not compromise neither the functionality of the overlay nor its structure. Accordingly, these experiments confirm the promising characteristics of bio-inspired solutions that can improve the robustness and reliability of distributed systems.

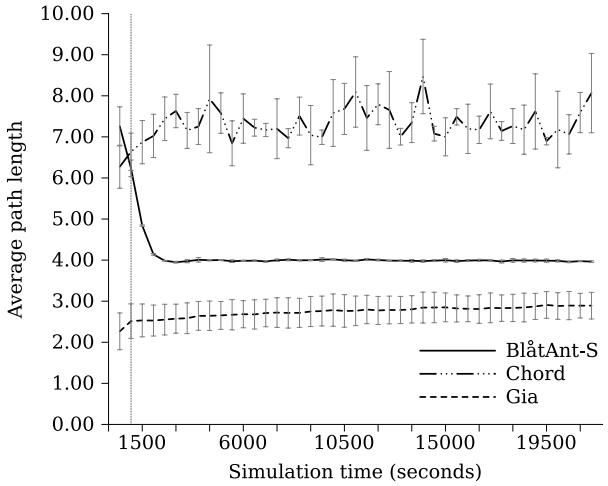

Fig. 24. Weibull, 1024 Nodes, 10% packet drop

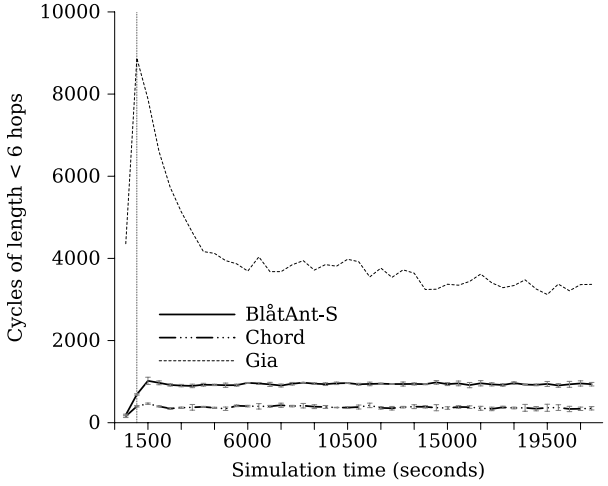

Fig. 25. Weibull, 1024 Nodes, 10% packet drop

C. Bandwidth consumption

Another important aspect of a network protocol is the generated overhead, namely the bandwidth consumed by maintenance messages. In this regard, we measured the traffic generated by each node in each scenario, and the results are reported in Tables IV and V for the reliable and unreliable communication scenarios respectively. Because the average traffic results is very consistent across all simulation runs we report the average standard deviation across nodes rather than across simulations. Among the three considered overlay protocols, GIA consumes the less traffic, both with 512 and with 1024 nodes. The topology adaptation mechanism is thus very efficient, although, as highlighted by previous results, it fails at maintaining a fully connected overlay. BLÅTANT-S produces the second highest overhead, nearly 4 times higher than GIA in the Weibull scenario on a reliable network. However, it also performs noticeably better than CHORD, which shows the highest bandwidth consumption.

As the network scales from 512 to 1024 nodes all protocols perform well, as the network overhead per node does not changes considerably. The only difference can be noticed with CHORD, and is presumably due to the increased variety of nodes in the finger table, which increases the cost of the stabilization procedure. It is interesting to note that GIA has the highest relative standard deviation, which surpasses in many scenarios 100% of the average. When an unreliable network is employed the bandwidth consumption of both CHORD and GIA decreases, while that of BLÅTANT-S increases. This result is due to the partitioning of the network that occours in both CHORD and GIA, which results in less maintenance messages sent to nodes that are thought dead by others. On the other hand, BLÅTANT-S maintains proper connectivity across the overlay, and the increase in traffic is in fact generated by recovery procedures.

Surprisingly, BLÅTANT-S generates more traffic in a stable overlay (without churn), because more messages are exchanged by nodes in order to optimize the topology. On the contrary, in churn situations less messages are sent, because agents can disappear on nodes that leave the overlay. Moreover, recovery procedures naturally contribute to reduce the diameter of the overlay, without requiring additional traffic. In conclusion, we can argue that BLÅTANT-S is able to cope well with both network of increasing size as well as churn and communication failures, without generating an excessive network overhead.

TABLE IV
BANDWIDTH CONSUMPTION, RELIABLE COMMUNICATION

Scenario	Avg Sent Bytes/s	St.Dev/Nodes
Weibull churn		
CHORD 512	78.10	15.31
CHORD 1024	87.11	21.77
GIA 512	6.24	6.02
GIA 1024	6.03	5.83
BLÅTANT-S 512	24.59	18.56
BLÅTANT-S 1024	24.40	18.95
Pareto churn		
CHORD 512	81.97	26.25
CHORD 1024	92.90	36.53
GIA 512	7.12	8.18
GIA 1024	7.33	8.27
BLÅTANT-S 512	20.56	16.90
BLÅTANT-S 1024	22.08	18.67
Random churn		
CHORD 512	78.16	19.19
CHORD 1024	86.74	19.07
GIA 512	6.67	7.01
GIA 1024	6.18	6.17
BLÅTANT-S 512	18.78	13.76
BLÅTANT-S 1024	23.08	18.18
No churn		
CHORD 512	77.98	16.86
CHORD 1024	86.74	20.37
GIA 512	7.48	5.59
GIA 1024	7.47	5.55
BLÅTANT-S 512	37.96	18.16
BLÅTANT-S 1024	38.66	19.90

TABLE V
BANDWIDTH CONSUMPTION, UNRELIABLE COMMUNICATION

Scenario	Avg Sent Bytes/s	St.Dev/Nodes
Weibull churn, 10% packet drop		
CHORD 512	31.65	16.74
CHORD 1024	36.71	15.99
GIA 512	4.08	5.99
GIA 1024	3.92	5.17
BLÅTANT-S 512	102.53	34.58
BLÅTANT-S 1024	105.02	35.89
Pareto churn, 10% packet drop		
CHORD 512	36.24	21.25
CHORD 1024	40.05	22.29
GIA 512	4.37	6.48
GIA 1024	4.75	6.76
BLÅTANT-S 512	85.35	46.61
BLÅTANT-S 1024	86.81	48.84
Random churn, 10% packet drop		
CHORD 512	38.45	17.52
CHORD 1024	38.06	16.67
GIA 512	4.00	5.27
GIA 1024	3.91	5.13
BLÅTANT-S 512	84.99	39.96
BLÅTANT-S 1024	102.04	37.17
No churn, 10% packet drop		
CHORD 512	35.13	4.79
CHORD 1024	34.55	5.92
GIA 512	3.06	3.92
GIA 1024	3.63	3.92
BLÅTANT-S 512	129.74	3.31
BLÅTANT-S 1024	131.69	3.47

VIII. CONCLUSIONS

In this paper we presented a detailed evaluation of BLÅTANT-S under churn and communication failures. By means of several simulation scenarios, it was demonstrated that BLÅTANT-S is able to cope well with typical churn patterns and with communication failures. The connectivity across the overlay as well as underlying optimization process are maintained even in highly dynamic scenarios. Thanks to the use of the OverSwarm framework, comparison between BLÅTANT-S and two traditional peer-to-peer protocols,

CHORD and GIA, was made possible. The obtained results confirm the data gathered in previous evaluations [4], and strengthen our belief that bio-inspired solutions can bring decisive benefits to distributed systems. On one hand, self-organized systems such as BLÅTANT-S ensure a high quality of service under normal circumstances, while making efficient use of the available network resources. On the other hand, the adaptiveness of a bio-inspired approach provides a robust and reliable solution that can overcome major communication failures. Our current research focuses on the evaluation of other bio-inspired solutions in order to draw hints and guidelines for their successful implementation in distributed systems.

IX. ACKNOWLEDGMENTS

This research is being carried out thanks to the financial support of the Swiss National Science Foundation, fellowship Nr. 134285.

REFERENCES

[1] Amos Brocco, Fulvio Frapolli, and Béat Hirsbrunner. Blatant: Bounding networks' diameter with a collaborative distributed algorithm. In *Sixth International Conference on Ant Colony Optimization and Swarm Intelligence*, pages 275–282. ANTS, Springer, September 2008.

[2] Amos Brocco, Fulvio Frapolli, and Béeat Hirsbrunner. Bounded diameter overlay construction: A self organized approach. In *IEEE Swarm Intel ligence Symposium (SIS 2009)*, April 2009.

[3] Amos Brocco, Apostolos Malatras, and Béat Hirsbrunner. Enabling efficient information discovery in a self-structured grid. *Future Generation Computer Systems, Elsevier*, 2010. Accepted.

[4] Amos Brocco. *Exploiting self-organization for the autonomic management of distributed systems*. PhD thesis, Department of Informatics, University of Fribourg, Switzerland, October 2010.

[5] Ion Stoica, Robert Morris, David Karger, Frans M. Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings SIGCOMM '01*, volume 31, New York, NY, USA, October 2001. ACM Press.

[6] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 407–418, New York, NY, USA, 2003. ACM.

[7] Amos Brocco. Overswarm: a simulation tool for biologically inspired peer-to-peer networks. Telematics Technical Reports 2011-4, Institute of Telematics, Karlsruhe Institute of Technology (KIT), aug 2011.

[8] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. OverSim: A flexible overlay network simulation framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007.

[9] Andras Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[10] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[11] Marco Dorigo, Mauro Birattari, Thomas Stützle, Université Libre, De Bruxelles, and Av F. D. Roosevelt. Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Comput. Intell. Mag*, 1:28–39, 2006.

[12] Zhenyun Zhuang, Yunhao Liu, Li Xiao, and Lionel M. Ni. Hybrid periodical flooding in unstructured peer-to-peer networks. *Parallel Processing, International Conference on*, 0:171, 2003.

[13] Vassilios V. Dimakopoulos and Evaggelia Pitoura. On the performance of flooding-based resource discovery. *IEEE Transactions on Parallel and Distributed Systems*, 17:1242–1252, 2006.

[14] H. Jiang and S. Jin. Exploiting dynamic querying like flooding techniques in unstructured peer-to-peer networks. In *Network Protocols, 2005. ICNP 2005. 13th IEEE International Conference on*, page 10 pp., nov. 2005.

[15] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *SIGOPS Oper. Syst. Rev.*, 41(5):2–7, 2007.

[16] Nabhendra Bisnik and Alhussein Abouzeid. Modeling and analysis of random walk search algorithms in p2p networks. In *HOT-P2P '05: Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 95–103, Washington, DC, USA, 2005. IEEE Computer Society.

[17] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, 2006.

[18] Luis Rodero-Merino, Antonio F. Anta, Luis López, and Vicent Cholvi. Performance of random walks in one-hop replication networks. *Computer Networks*, October 2009.

[19] *Peer-to-peer architecture case study: Gnutella network*, August 2001.

[20] Luis Rodero-merino, Antonio Fernndez, Luis Lpez, and Vicent Cholvi. A topology self-adaptation mechanism for efficient resource location. In *In ISPA 2006*, pages 660–671. Springer-Verlag, 2006.

[21] A. Ma. Caida: tools: measurement: skitter, June 2008. accessed on 2008-06-08.

[22] Zhongmei Yao, Derek Leonard, Xiaoming Wang, and Dmitri Loguinov. Modeling heterogeneous user churn and local resilience of unstructured p2p networks. In *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 32–41, Washington, DC, USA, 2006. IEEE Computer Society.

APPENDIX

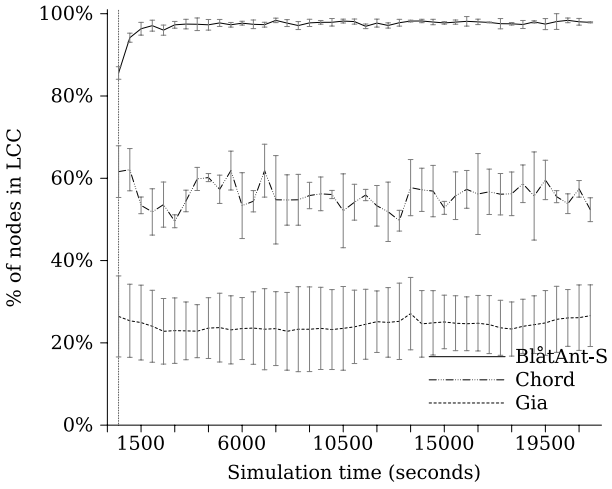

Fig. 26. Random, 512 Nodes, 10% drop

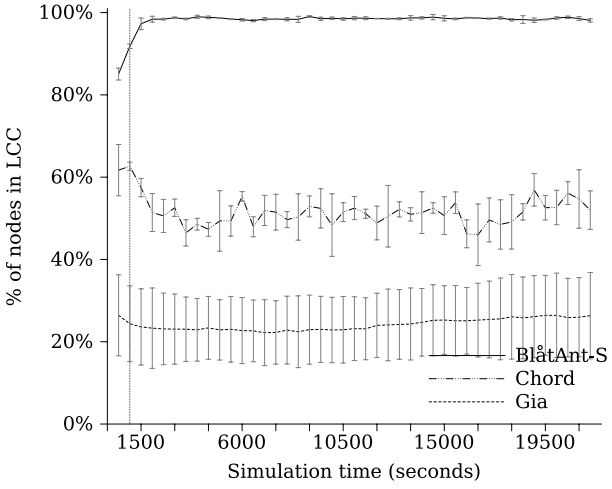

Fig. 27. Random, 1024 Nodes, 10% drop

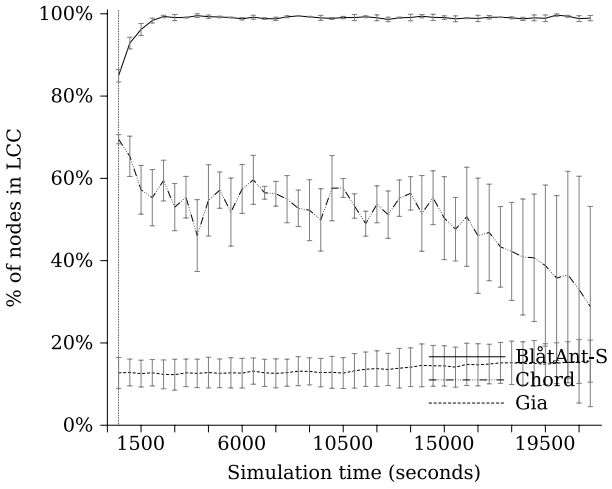

Fig. 28. Pareto, 512 Nodes, 10% drop

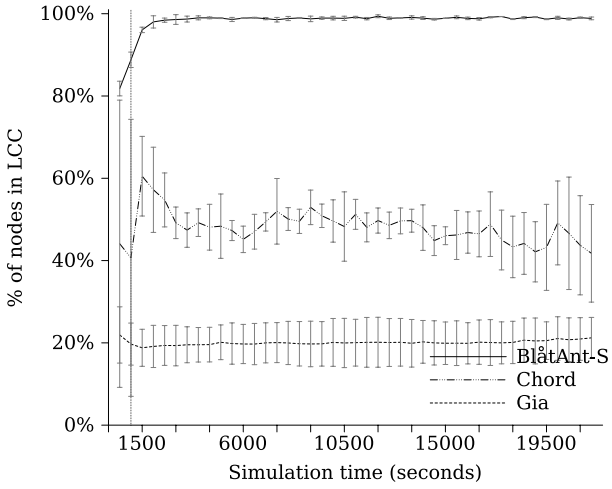

Fig. 29. Pareto, 1024 Nodes, 10% drop

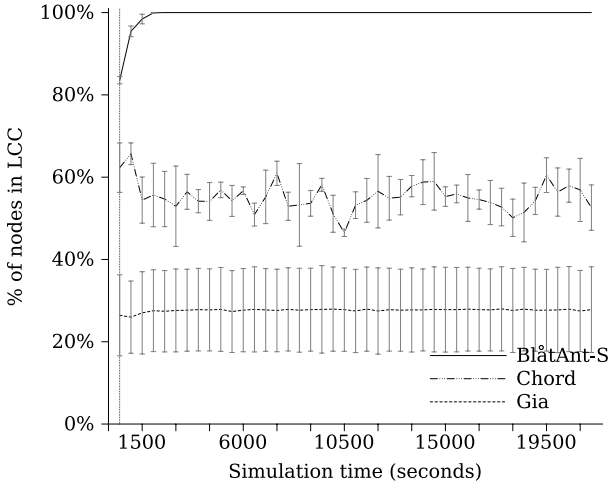

Fig. 30. No churn, 512 Nodes, 10% drop

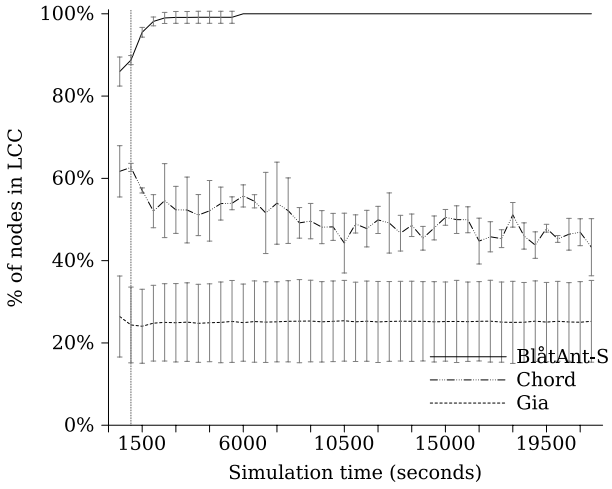

Fig. 31. No churn, 1024 Nodes, 10% drop

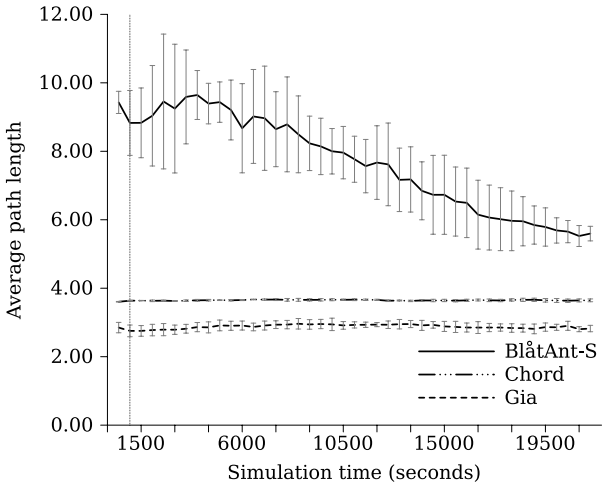

Fig. 32. Random, 512 Nodes

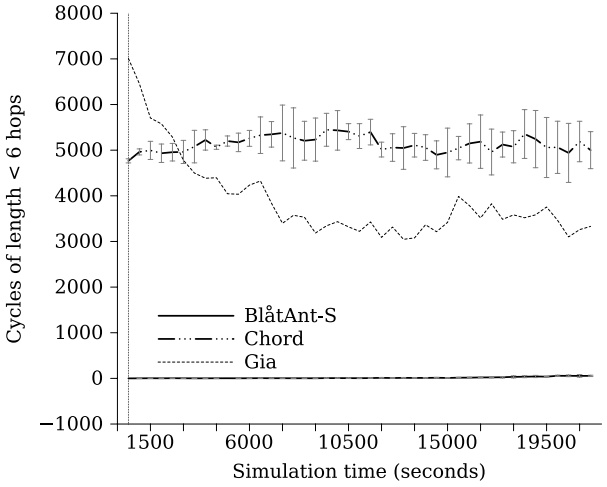

Fig. 33. Random, 512 Nodes

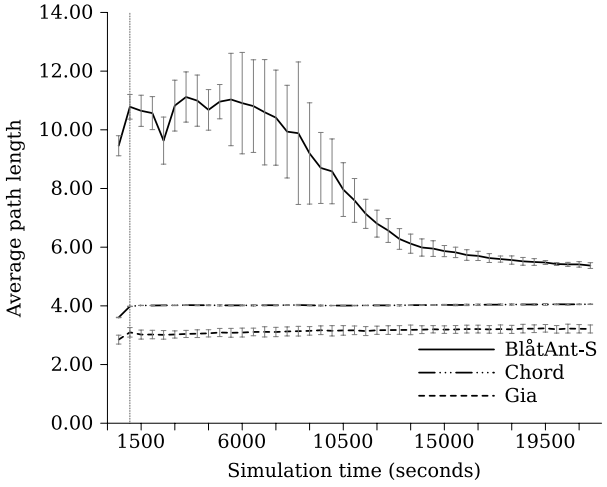

Fig. 34. Random, 1024 Nodes

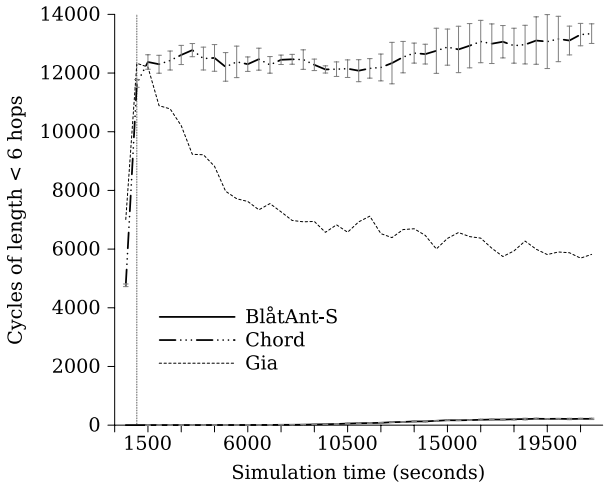

Fig. 35. Random, 1024 Nodes

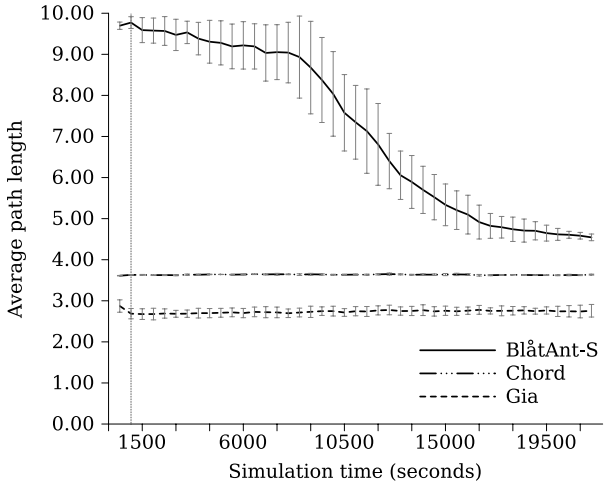

Fig. 36. Pareto, 512 Nodes

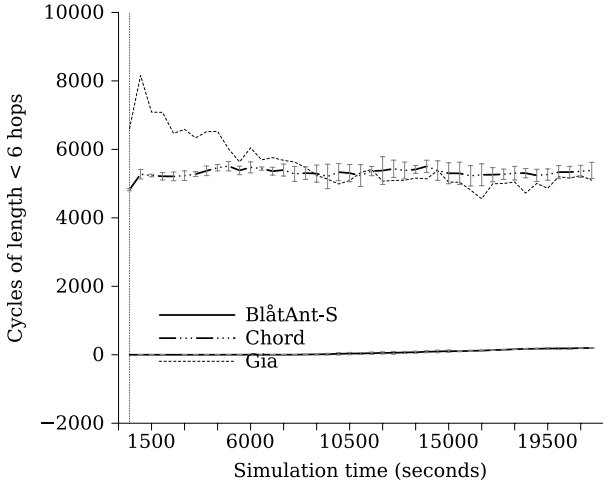

Fig. 37. Pareto, 512 Nodes

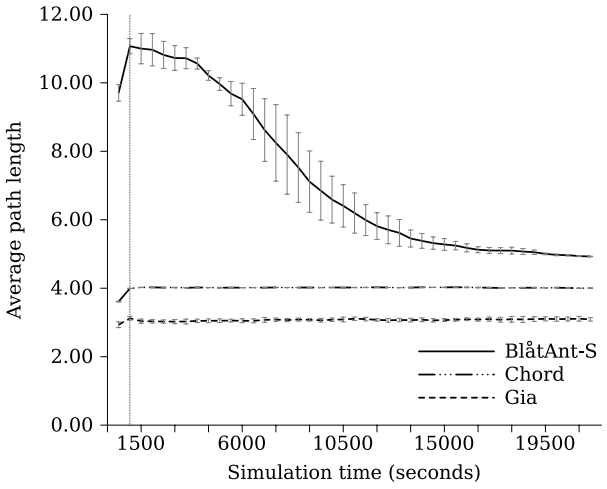

Fig. 38. Pareto, 1024 Nodes

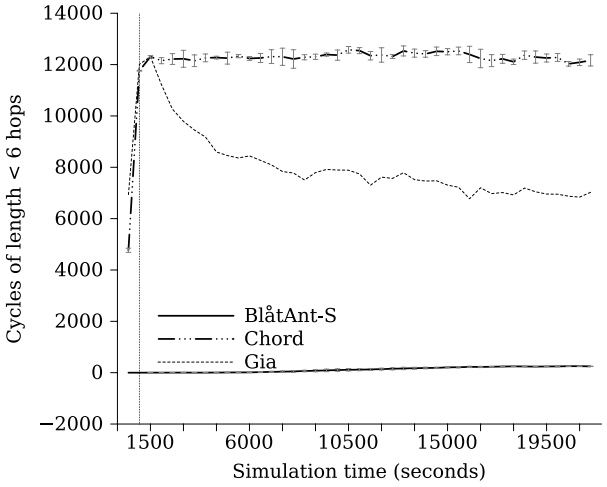

Fig. 39. Pareto, 1024 Nodes

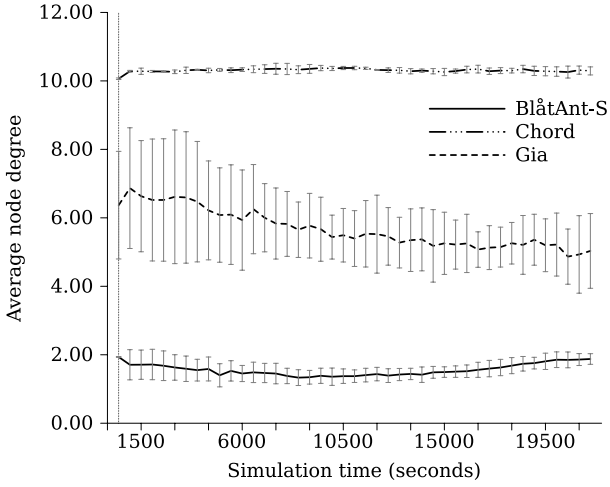

Fig. 40. Random, 512 Nodes

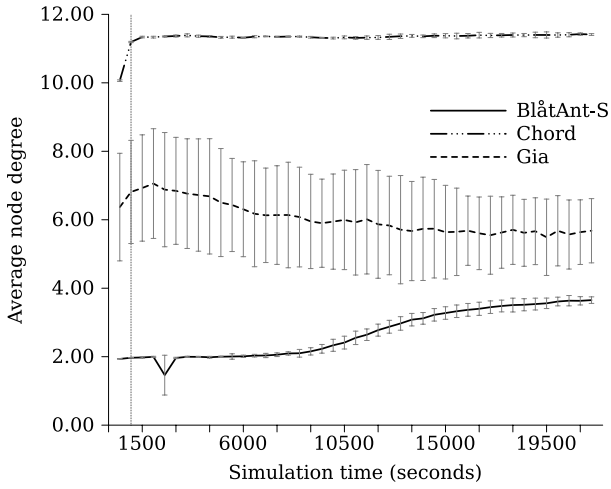

Fig. 41. Random, 1024 Nodes

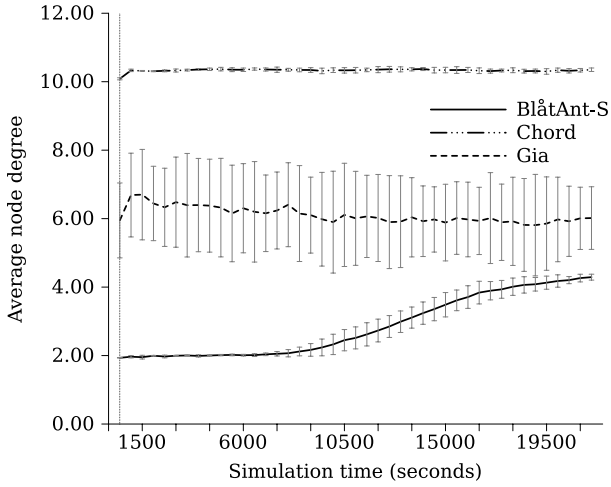

Fig. 42. Pareto, 512 Nodes

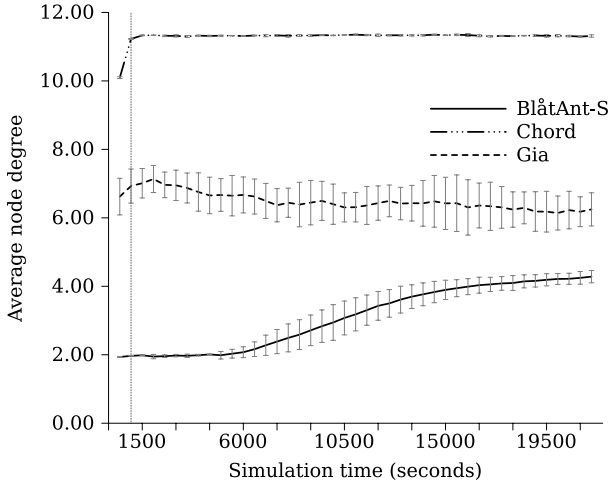

Fig. 43. Pareto, 1024 Nodes

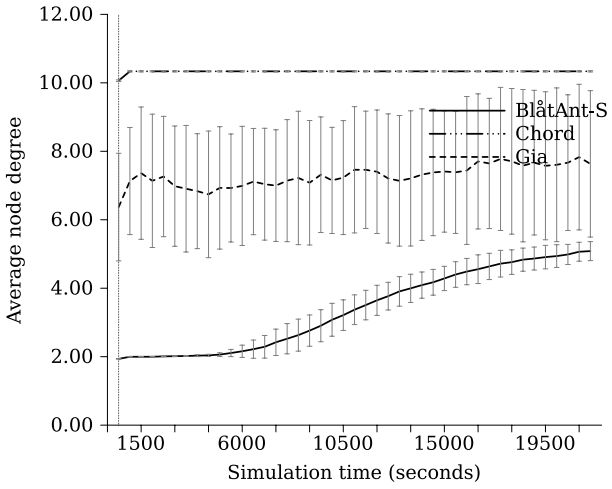

Fig. 44. No churn, 512 Nodes

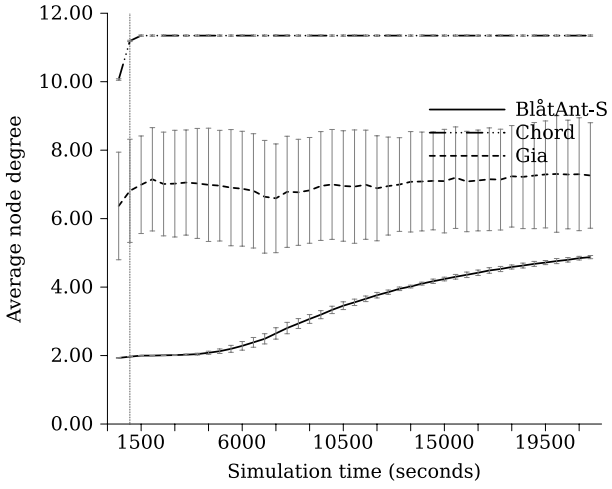

Fig. 45. No churn, 1024 Nodes

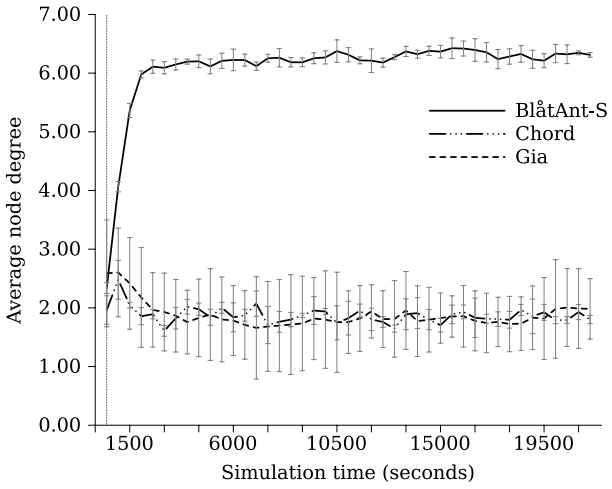

Fig. 46. Random, 512 Nodes, 10% drop

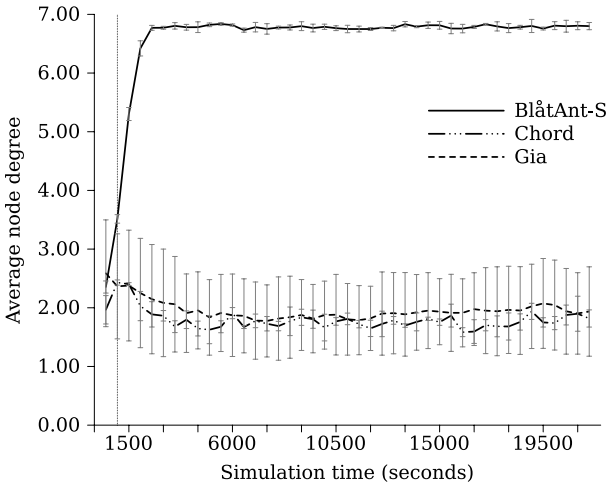

Fig. 47. Random, 1024 Nodes, 10% drop

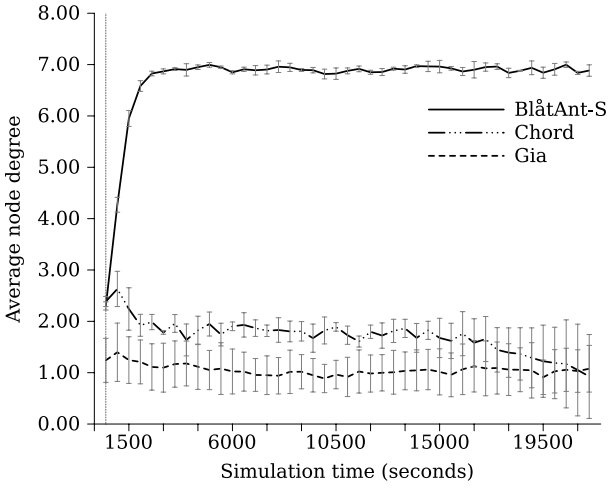

Fig. 48. Pareto, 512 Nodes, 10% drop

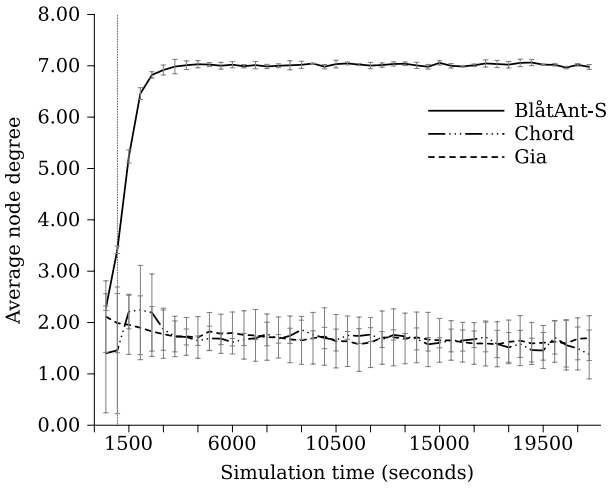

Fig. 49. Pareto, 1024 Nodes, 10% drop

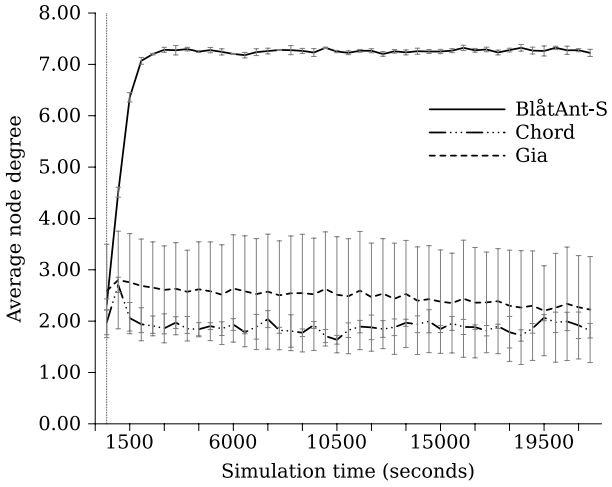

Fig. 50. No churn, 512 Nodes, 10% drop

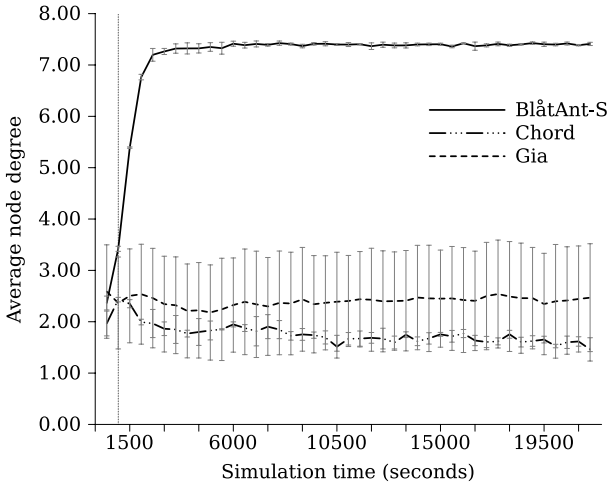

Fig. 51. No churn, 1024 Nodes, 10% drop

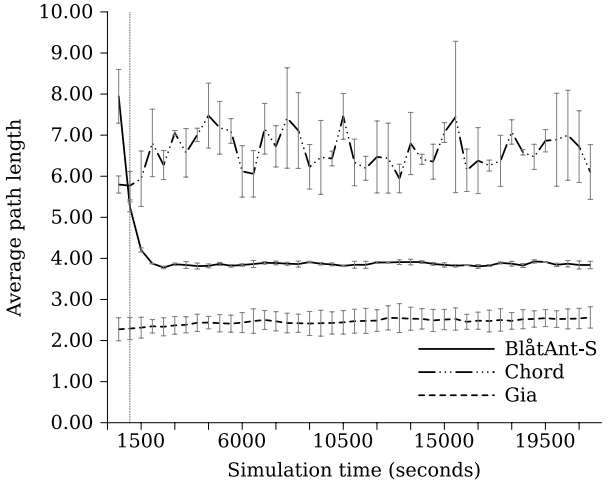

Fig. 52. Random, 512 Nodes, 10% packet drop

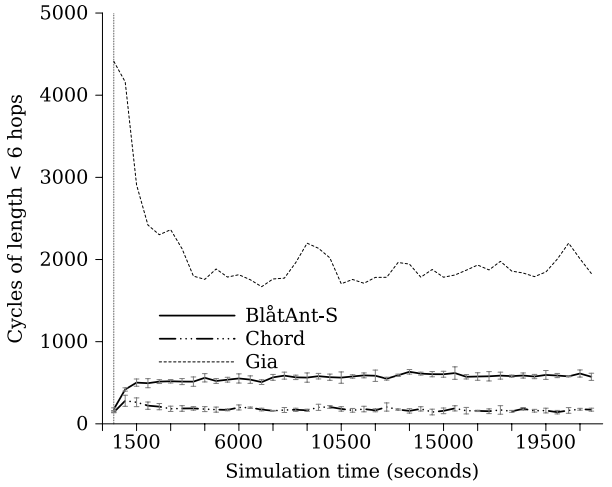

Fig. 53. Random, 512 Nodes, 10% packet drop

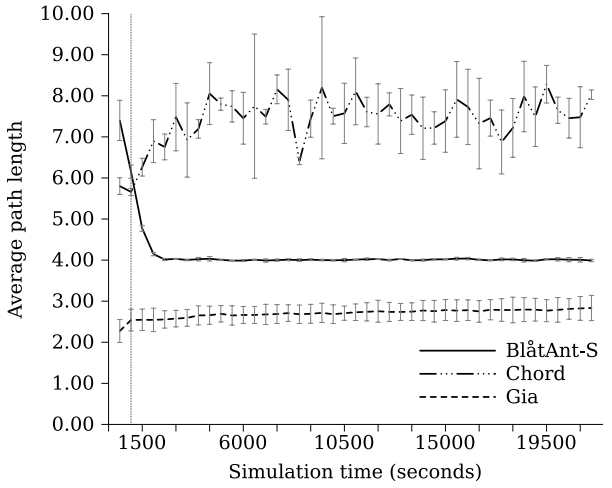

Fig. 54. Random, 1024 Nodes, 10% packet drop

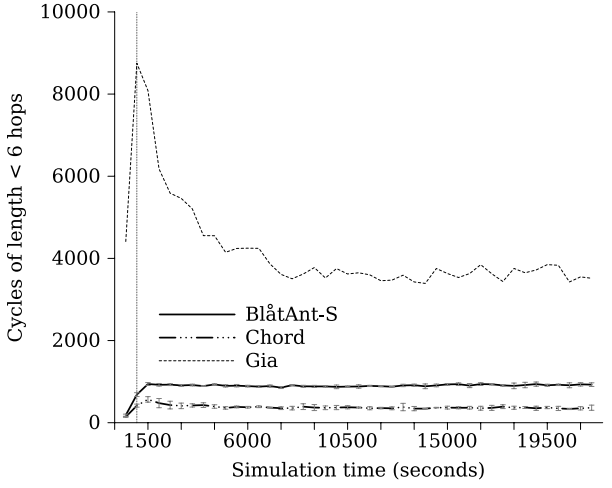

Fig. 55. Random, 1024 Nodes, 10% packet drop

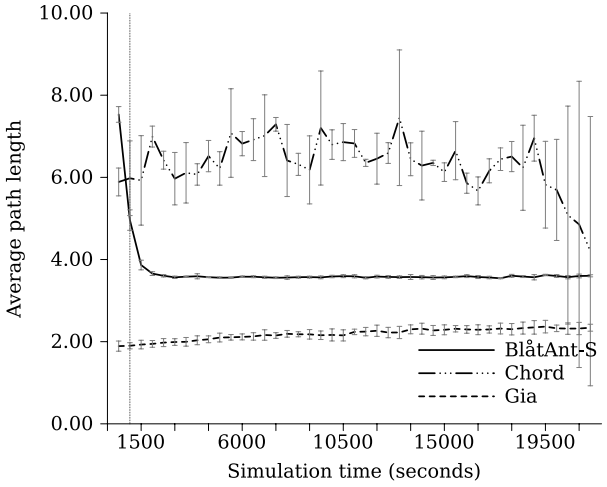

Fig. 56. Pareto, 512 Nodes, 10% packet drop

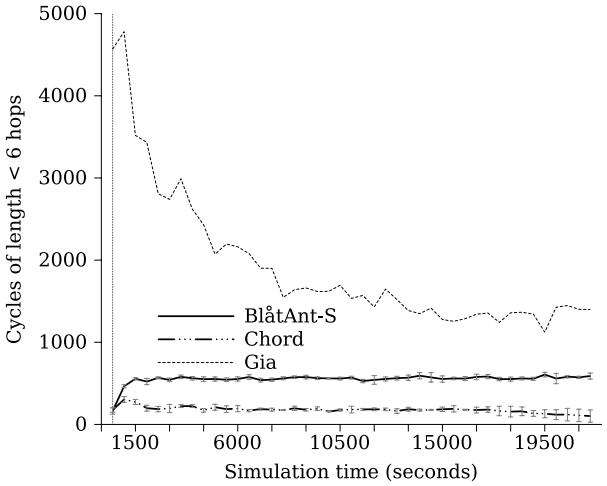

Fig. 57. Pareto, 512 Nodes, 10% packet drop

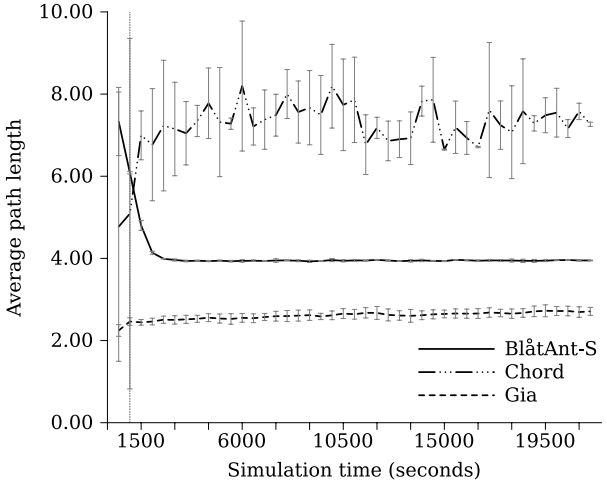

Fig. 58. Pareto, 1024 Nodes, 10% packet drop

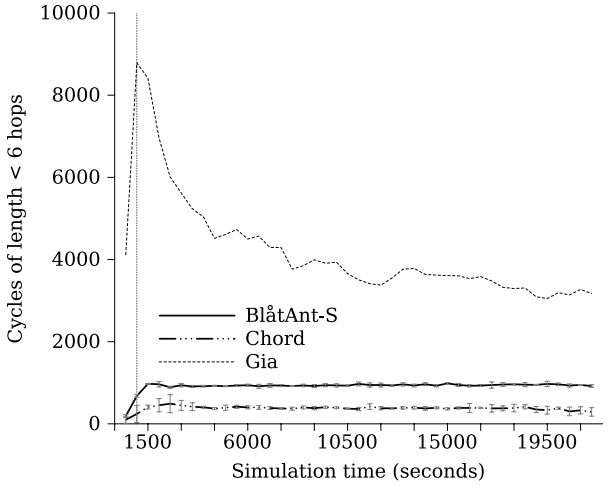

Fig. 59. Pareto, 1024 Nodes, 10% packet drop

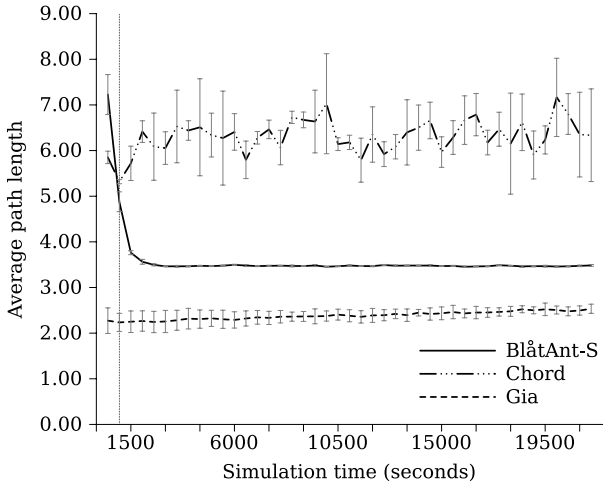

Fig. 60. No churn, 512 Nodes, 10% packet drop

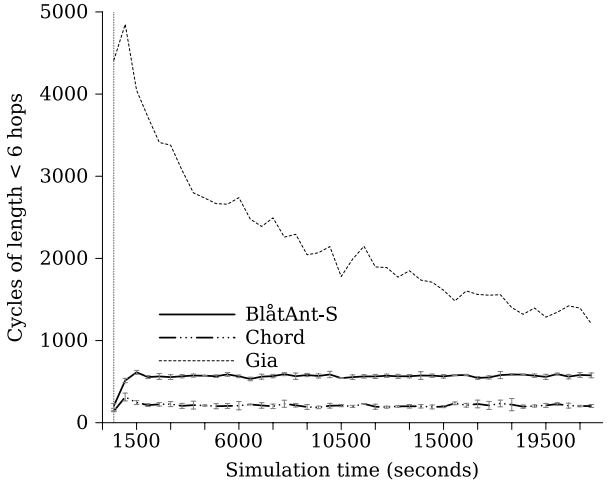

Fig. 61. No churn, 512 Nodes, 10% packet drop

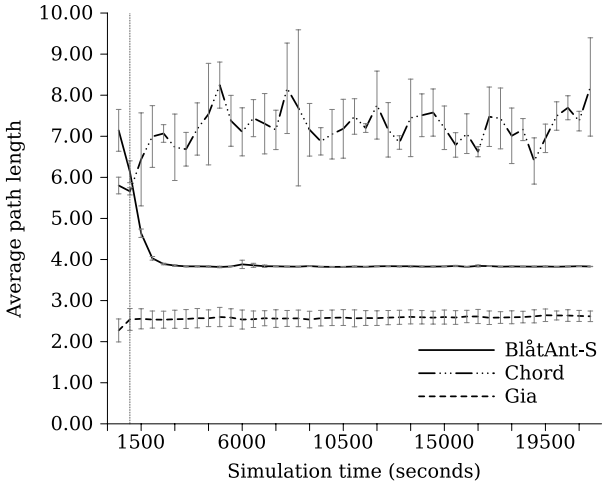

Fig. 62. No churn, 1024 Nodes, 10% packet drop

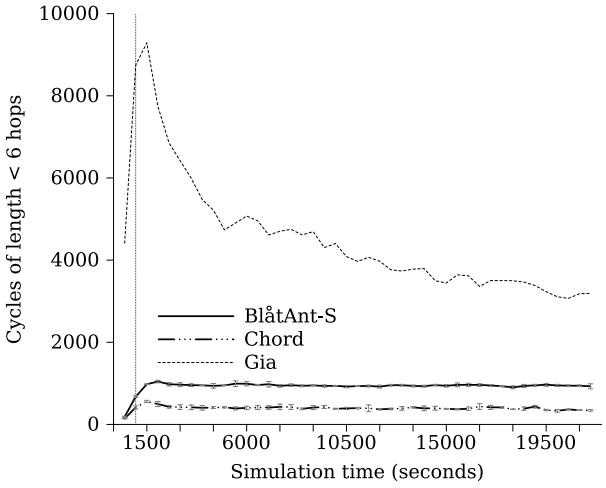

Fig. 63. No churn, 1024 Nodes, 10% packet drop

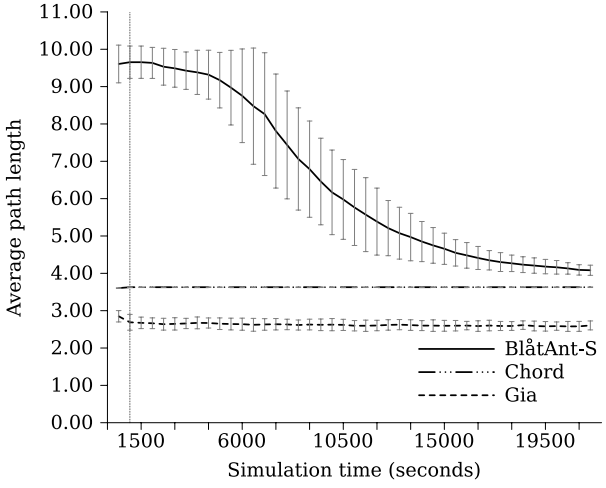

Fig. 64. No churn, 512 Nodes

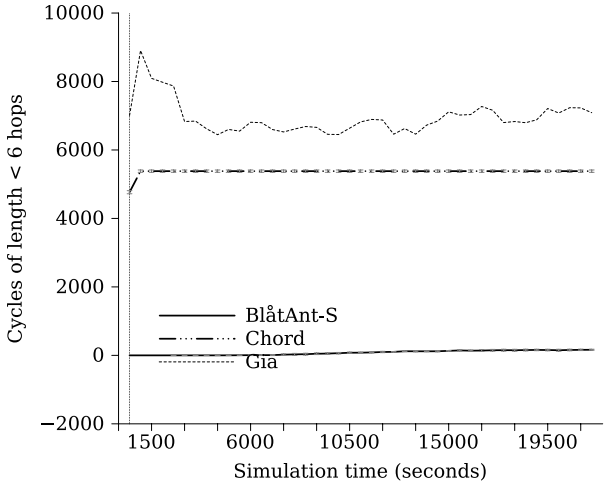

Fig. 65. No churn, 512 Nodes

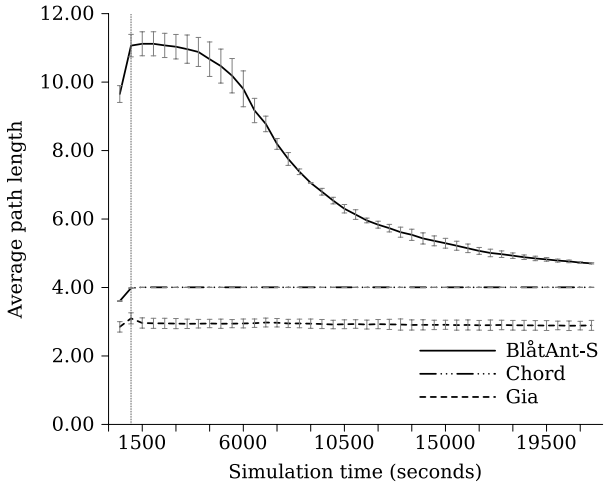

Fig. 66. No churn, 1024 Nodes

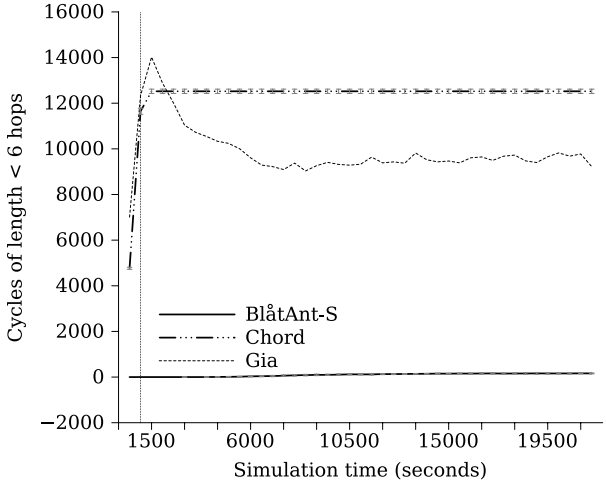

Fig. 67. No churn, 1024 Nodes