Sobe and Elmenreich *Complex Adaptive Systems Modeling* 2013, **1**:13
http://www.casmodeling.com/content/1/1/13

Complex Adaptive Systems Modeling
a SpringerOpen Journal

RESEARCH Open Access

Replication and replacement in dynamic delivery networks

Anita Sobe[1,2*] and Wilfried Elmenreich[1,3]

*Correspondence:
anita.sobe@unine.ch
[1]Institute of Networked and Embedded Systems/Lakeside Labs, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
[2]Computer Science Department, University of Neuchâtel, Neuchâtel, Switzerland
Full list of author information is available at the end of the article

Abstract

Purpose: Content delivery in dynamic networks is a challenging task, because paths may change during delivery and content might get lost. Replication is a typical measure to increase robustness and performance.

Method: In previous work we proposed a hormone-based algorithm that delivers content, and optimizes the distribution of replicas. Clients express demands by creating hormones that will be released to the network. The corresponding resources are attracted by this hormone and travel towards a higher hormone concentration. This leads to a placement of content close to their most frequent requesters. In addition to that the hormone-based delivery requires an appropriate replication and clean-up strategy to balance the replicas throughout the network without exceeding the nodes' storage limits or the networks communication capacity.

Results: We examine different combinations of replication and replacement strategies and evaluate them in realistic scenarios involving node failure and networks of different size and structure.

Conclusion: Results show that it is necessary to match the replication mechanisms with the clean-up mechanism and that the local hormone information can be used to improve the clean-up decision.

Keywords: Self-organization, Replication, Storage balancing, Artificial hormone system, Multimedia distribution

Background

Introduction

In peer-to-peer networks each node can act as a client or server for the other nodes in the network, allowing shared access to various resources such as files, peripherals, and sensors without the need for a central server. We consider peer-to-peer networks, where content is dynamically stored at different nodes and delivered to the client on request. Depending on the request patterns in the network, stored content is migrated from one node to another, replicated or dissolved. Such a dynamic delivery system needs to implement two functions: the primary function is to provide the clients with the requested content with short delay. The secondary function is to optimize the placement of replicas in the network to minimize network delay for future requests.

When considering limited communication capabilities, the fulfillment of the two functions mutually influence each other. The placement optimization will increase the network load for some time and therefore worsen the system's responsiveness to client requests. Over time, the re-arrangement and replication of content, however, will improve the delivery performance and reduce network load. In applications with smart sensors or mobile devices, the system operation is also significantly affected by the limited storage of embedded devices. Thus, it is necessary to clean-up unused replicas or to balance the storage of content according to the available storage on the nodes. Moreover, we assume that the network is dynamic and therefore it is impossible to build global knowledge about all nodes and the stored content. The dynamics of the network stem from new generation of content/deletion of content, devices joining/leaving the network, and changing locations of a device due to user mobility. In contrast to a centralized solution, this makes it hard to ensure that for each content unit at least one copy remains in the system at all time. Since lost content can usually not easily be recovered this is an important requirement for most content storage networks. In order to solve the distributed multi-optimization problem for storage balancing, request delay, and network load (within the network) we propose a distributed self-organizing and self-adapting replication and replacement strategy.

The hormone-based delivery algorithm (Sobe et al. 2010) is a promising candidate for delivering and distributing content in large networks. It assumes only local knowledge at each node. The central idea of this algorithm is to use artificial hormones to request transport and replication of content. The hormone-based algorithm is managed without global knowledge and consists of two parts: (1) Requests for content are represented by hormones. Hormones are created on an arriving request, are diffused to neighbors and evaporate over time; (2) hormones trigger the transport of corresponding content, which is done hop-by-hop. A further important mechanism is replication, which is not only a measure of robustness; the algorithm also exploits replication by placing replicas on the transport path. This reduces the search space.

In contrast to ant algorithms (Dorigo et al. 2006), which can be used for robust routing in dynamic networks, the hormone approach provides a means for optimizing the content placement in the network. In the endocrine system of higher mammals, hormones are created and travel around the body where they lead to specific actions in target cells. The same hormone might lead to different actions, depending on the type of the target cells (Xu et al. 2011). This is also the main principle of the hormone-based delivery system. Nodes in the network represent cells that are able to create, consume and forward hormones. As a reaction to hormones a node can consume, replicate, forward content or even do nothing. A further difference is that we do not assume static content location, such as the food source in ant-based systems. In our system content can replicate and move towards the receiver.

It is the purpose of this paper to examine different replica replacement strategies for large distributed systems to balance the storage of nodes. The results of our evaluation show that the selection of the appropriate clean-up mechanism, i.e., the particular strategy for removing or migrating units to ensure sufficient storage capability at a node to fulfill future requests has a crucial effect on the network performance. An initial summary of our results have been made available as a technical report (Sobe et al. 2011b) to enable discussion.

The paper is structured as follows: In the following section we introduce related work on replication strategies for bio-inspired distribution networks. To emphasize the relevance of our problem statement, we elaborate the context of the targeted problem followed by a detailed description of the hormone-based delivery and replication approach. A special focus is given on the different replacement mechanisms. We further consider the constraint that a clean-up action must never delete the last instance of a unit. The described strategies are implemented in a simulation model of a network of multimedia servers with limited storage and consumers requesting content according to a preference model. The simulation results allow for a quantitative comparison of the efficiency of the different variants. We evaluated realistic scenarios involving node churn and networks of different size and structure. The conclusion summarizes our recommendation for effective replication and replacement strategies in dynamic content delivery networks.

Related work

Typically, the popularity of content is dynamic and replicas of content with decreasing popularity have to be handled to balance the storage. This can be done by introducing clean-up mechanisms; however, the challenge is to identify unnecessary replicas.

Lee et al. describe in (Lee et al. 2008a) a utility based replication scheme, named SmartPin for pre-recorded IPTV content in peer-to-peer streaming systems. The IPTV content is divided into variable length segments. Depending on the utility, content and/or its metadata are actively replicated to other peers. E.g., for intermediate utility content, only the metadata containing the original place of the content is replicated. The utility depends on the popularity of the content, its benefit and its costs. As benefit the authors define the perceived quality and the required bandwidth, whereas the costs regard the delivery and storage costs (Lee et al. 2008b). To increase the content availability the minimum number of needed replicas is calculated centrally.

A utility function measuring the importance of content is a first step to handle the dynamics of user preferences. Nonetheless, the utility depends on the popularity, which is not easy to derive in a system without global knowledge. The authors do not mention what happens to the replicas if the popularity of the content changes.

The author of (Rong 2008) proposes another replication scheme for peer-to-peer video systems that adapts the number of replicas according to their utilization rate. Temporary popular videos are replicated more often and the replicas are destroyed when the number of requests for the video decreases. The number of replicas is handled by a central coordinator. In a system with local knowledge only a more dynamic solution has to be provided.

In (Herrmann 2007) an example for service replica placement in ad-hoc grids is shown, where the cost of a request is measured by the number of message transmissions per time unit to serve it. The goal is to minimize those costs without global coordination. The author defines a parameter ρ which describes the coverage radius of clients. If the number of requests for this service is below a given threshold, the service replica removes itself from the system. Service and content replication is comparable, however, in this work services are independent from each other. If a composition of services is provided the cost function would have to be adapted to a more flexible system.

Forestiero et al. proposed a descriptor sorting and replication algorithm for peer-to-peer grids called *Antares* in (Forestiero et al. 2007). This approach is not only ant inspired, but can also be categorized as *brood sorting* such as described in (Mamei et al. 2006). The descriptors are bit strings encoded by locality sensitive hash algorithms (see Kulis and Grauman 2009). In contrast to standard hash algorithms, similar content results in similar hash values. This helps to sort similar content based on its hash value. The sorting is done by ant-inspired agents that travel the grid and pick and drop descriptors. Agents operate in two modes, copy and move. In copy mode the agent creates a replica of the descriptor, whereas in move mode not. The transition between these states is pheromone based. In the beginning, each agent starts in the copy mode. At some point, when the resources are better sorted, the activity of the agent decreases. Then, the agent starts to increase its pheromone level until a given threshold and subsequently switches to the move mode.

It is interesting that Antares has also been adapted to support QoS-based picks and drops (Forestiero et al. 2008). Descriptors with better QoS are more likely to be replicated. The QoS is described as a non-negative real value, where higher values describe higher QoS. Analogous to the pick probability the drop probability increases if a descriptor with high QoS is carried by an agent. Although the placement of replicas is handled by Antares, the number of replicas is never regulated. An extension of the model by introducing a *delete agent* could be done by using one of the mechanisms described in the current paper.

In (Sobe 2012) we give a more detailed overview and some more related work on these topics.

Context

Our evaluation is based on a scenario of a social event like a triathlon. Visitors follow such an event along a very large area. To see what is happening around the track they have to rely on videos provided by the organizers on video walls. The possibility of querying content such as *"I want an overview of interesting parts of the last 30 minutes"* or *"I want to see more of the athlete with no. 1234"* is missing. Visitors produce masses of multimedia data, but there is no specific possibility to exploit this data for live sharing with other visitors. Visitors should be able to create their individual presentations depending on their interests (Böszörmenyi et al. 2011).

We define the term multimedia unit, which can be either a photo or a short video sequence, e.g., generated at a live sports event by a visitor or a picture. We further assume dynamic access patterns. Users can "compose" their presentations consisting of a number of different units in their preferred order. In (Sobe et al. 2010) we also support parallel presentations, such as overviews in a split screen. Such access patterns allow flexibility, but also introduce complexity, because the typical pre-defined video sequence as known from movies is not existing anymore.

The triathlon scenario further shows that content gets more and more important both on the consumer and on the producer side. In this scenario visitors produce multimedia content all around the area, and should also be able to consume anything (multimedia content), anytime and anywhere they want. However, the transport of content is still limited to traditional, mostly static, delivery methods. Future Internet discussions such as described by Hausheer et al. in (Hausheer et al. 2009) show that flexible solutions for delivery are needed.

The used delivery algorithm is capable of handling this complexity relying on simple local information. The algorithm is inspired by two existing bio-inspired approaches. A specific ant-based application for search in peer-to-peer networks was introduced as SemAnt (Michlmayr 2007). The second approach is an artificial hormone-based agreement for task allocation introduced by Brinkschulte et al. in (Brinkschulte et al. 2007).

We adopted the keyword search from SemAnt and introduce one type of hormone by keyword. The hormone value expresses the current demand (the *goodness*, as adopted from Brinkschulte et al.) for the corresponding keyword. SemAnt assumes static content locations, which keeps the search space very large. To reduce the search space we replicate content and exploit the unstructured overlay by letting the content travel towards higher levels of its corresponding hormone. This allows intermediate nodes to decide if the traveling unit is needed for future requests, thus predicting places to reduce search space and delivery costs.

This work relies on a bio-inspired delivery algorithm introduced in (Sobe et al. 2010). This basic version of the algorithm only considers the hormone creation, diffusion and transport of content based on hormones. Different replication mechanisms have been evaluated in (Sobe et al. 2011a).

Since peers are not likely to provide unlimited storage for other peers and in a dynamic system the popularity of units changes, in this paper we investigate different measures to efficiently balance the storage of the peers. We periodically apply different strategies if a certain storage level is reached. We compare LRU (Least Recently Used), LFU (Least Frequently Used) and a hormone-based clean-up. We show that the chosen clean-up mechanism has an impact on the delivery performance and that the system is still robust against peer failure.

Method

Hormone-based delivery and replication

The hormone-based delivery approach introduced in (Sobe et al. 2010) involves three components: hormone creation, hormone diffusion, and the behavior of units in presence of a corresponding hormone. The distributed, self-organizing nature of the approach assists with handling the complexity of requests and the search for units in the network with comparably simple decision algorithms based on local knowledge.

Upon request of a particular unit, a corresponding hormone is issued at the requesting node. The hormone is represented as a real value – the current value of a hormone is the *hormone level*. The hormone is diffused via the neighboring nodes to the network, creating a hormone gradient towards the requesting node.

The diffusion of hormones depends on the network structure – a node only a few hops away from the requesting node will get more hormones than a more distant node.

Figure 1 shows an example of the diffusion of hormones from a requesting node P_1. Assume a network of seven nodes P_1 to P_7. Links have different quality, which is indicated by line thickness. Node P_1 creates hormones (`total`) for the requested unit. To create a hormone gradient P_1 splits the available hormones and reserves a part for diffusion to neighbors (`to_diffuse`). How much each neighbor gets from `to_diffuse` depends on the quality of the link to it. Thus, P_2 gets the highest rate and P_4 the lowest rate. When the neighbors get the hormones, they forward again a part of it. The forwarding is repeated until a node is reached where the requested content is located (here, P_7). The unit can now move hop-by-hop towards the highest hormone concentration, which is found at the requester.

Beyond this simple example we further implemented continuous hormone creation. As long as a request is unfulfilled, the hormone level is raised periodically, thus increasing

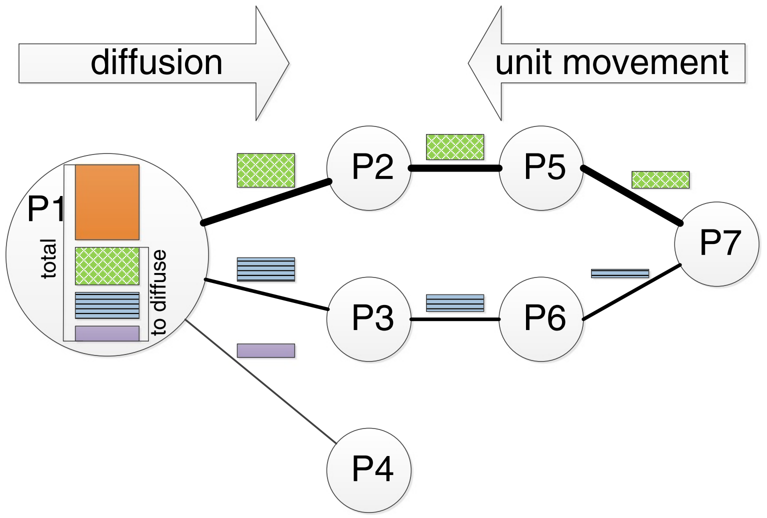

Figure 1 Example of the artificial hormone diffusion. Interplay of diffusion and unit movement in hormone-based delivery.

the concentration of hormones at the sender end, encouraging diffusion of hormones towards the receiver, until the request is fulfilled.

A unit that corresponds to the given hormone will move towards the nodes with the most of that hormone and relies on the diffusion mechanism to choose currently the best path. In order to reduce attracting multiple copies of a unit, the diffusion of a hormone is stopped if the respective unit already resides on the node. The hormone-based delivery creates a feedback loop between network conditions. The stability of this feedback loop depends on the parameter settings, which are discussed in the following section. Multiple requests for different units lead to a different set of hormones being handled in parallel by the network. Requests for the same unit result in a superimposed hormone landscape for that unit. In this case, a unit might be attracted by two hormone trails. Without replication, the unit must move to different requesters in order. The requester that receives the unit first is determined by the strength of hormone reaching the unit (from the requester). In order to avoid such detours, an intelligent replication mechanism has to take care of this issue.

Parameter settings

The proposed algorithm is self-organizing and depends on a number of configuration parameters, which are listed in Table 1. The most important ones are η_0 and η, α and ϵ, because they are defining how many hormones are created, diffused and evaporated per time step. η_0 controls the hormones created at the issuing of the request, η defines the hormone per time unit that are added at the requesting node until the request is fulfilled. α is the percentage of hormones that can be diffused to neighbors (the result of $\alpha * total$ corresponds to `to_diffuse` in Figure 1). The evaporation value ϵ will be subtracted for reducing the hormones on alternative paths. These parameters need to be tuned dependent on each other. E.g., if η_0 and η are low and the evaporation value ϵ is high, the movement of units is limited to a fewer number of hops. The greater the amount of hormones created, the further the hormones can travel and thus the search

Table 1 Parameters to configure at system startup

ID	Explanation
η_0	Hormone strength of a unit at new request
η	Increase of hormone after each time step by the requester
α	Percentage of hormones to be forwarded to the neighbors
ϵ	Hormone evaporation value
t	Significance threshold for hormones
m	Minimum hormone strength difference to move unit

space increases. Further parameters regard the minimum hormone strength difference to move a unit m, which controls the mobility of units. If m is high, the units need a higher hormone concentration to move to a neighbor, leading to a longer waiting time for the requester. t is the minimum hormone strength, if a hormone value is below this threshold it is considered as insignificant and can be deleted. In general, the parameter settings are essential for the algorithm to work and their inter-dependencies and combinations make it hard to tweak them manually. We therefore optimize them using an evolutionary algorithm as described in (Elmenreich and Klingler 2007).

Initially, the algorithm creates a random population of parameters. It then uses elite selection for creating the next generation. The candidates are sorted according to their fitness and the best x candidates are chosen. These candidates propagate to the next generation. To reach the same population size as the last generation, the rest of the slots are reserved for mutation, crossover and new candidates. For mutation and crossover random elite candidates are chosen. Finally, random new candidates are added to the population.

Replication mechanisms

We target content delivery systems that set up on top of unstructured peer-to-peer overlays and we investigated in (Sobe et al. 2010) that efficient replication mechanisms are necessary, thus we compared existing and proposed replication mechanisms in (Sobe et al. 2011a). The following categorization is based on (Androutsellis-Theotokis and Spinellis 2004) and (Yamamoto et al. 2005).

Existing replication mechanisms

Owner replication

The content is replicated at the requester's node (Lv et al. 2002) and is also called passive replication. Typically, this replication technique is used in file sharing systems based on BitTorrent (Cohen 2003). BitTorrent supports direct download, if a resource is found it is copied to the requester. Only nodes that are interested get the resource.

Path replication

In a multi-hop network where content is not transported directly such as in Freenet (Clarke et al. 2001), it is possible to cache one replica of the content at each intermediate node. Since the intermediate nodes are acting as caches, path replication is also called cache-based replication. It is assumed that intermediate nodes provide storage space for replicas even if they are not interested in the content. Path replication leads to a high number of unused replicas.

Therefore, an improved approach replicates the content on an intermediate node according to a fixed replication rate (*path random replication* (Yamamoto et al. 2005)). The advantage of this approach is a compromise between a higher replica usage and limited hop distance to other replicas. The difficulty of this approach is to specify a suitable replication rate for each file in advance, which is hard if the files are not known at system startup.

An alternative is to specify a node specific replication probability, where nodes decide ad-hoc if a file is replicated or not. The replication probability is dependent on the peer's resource status and optionally refers to the replication rate, too. The authors in (Yamamoto et al. 2005) refer this strategy to as *path adaptive replication.*

Active replication

The goal is to place the right number of replicas at the right locations *before* they are requested. Researchers investigated the optimal number of replicas in the context of robustness. In (Cohen and Shenker 2002) and (Lv et al. 2002) the authors investigate *random, proportional, and square root replication.* When applying random replication a uniform number of replicas for each object are created. Proportional replication creates replicas proportional to their query rate. The authors showed that square root replication determines the optimal replication rate r_i for object i, which is calculated as $r_i = \lambda\sqrt{q_i}$, with $\lambda = R/(\sum_i \sqrt{q_i})$, where q is the query rate and R is the number of object replicas in the system. Square-root replication does not consider the location of replicas. All strategies require global knowledge on the number of currently existing replicas and the current query rate for each of the replicas.

To reach square-root replication with limited knowledge researchers proposed *Pull-then-Push Replication* introduced in (Leontiadis et al. 2006). The first phase of this method regards the search of the content, with any existing algorithm. The second phase regards the replication of content to the neighbor nodes. To reach square root replication, the authors suggest that for the pull and push phase the same algorithms are used, because the number of replicas should be equal to the number of nodes visited during search. The authors evaluated typical algorithms, such as flooding and random walks. Their focus is on robustness even on update situations. As multimedia content is usually not updated after creation and this algorithm only considers the number and not the location of content, this approach is out of scope of this work.

Local knowledge replication mechanisms

Although a unit is delivered hop-by-hop the basis for the following replication mechanisms is owner replication, since the units are consumed for some time at the requester and therefore need to be replicated to be further usable by other nodes. Units replicated at the requester can only be supportive for the immediate neighborhood. To serve future requests, replicas should also be created on the delivery path. If the hormone concentration of a neighbor attracts a stored unit, the peer has to decide whether to move or to copy the unit. The simplest solution would be to apply path replication, but then the utilization of replicas would drop and the storage space is not used efficiently. Therefore, the goal is to find a replication mechanism that balances replica utilization and delay without the need of global information.

Simple hormone

If a unit is requested by peers from different parts of the network, the unit has to move first to one requester and afterwards to the other requester. This can lead to long traveling paths, which can be avoided by replicating a unit if more than one neighbor holds hormones for it. Note that it is not possible to differentiate if hormones on the neighbor are created by different peers. Thus, it is possible that unnecessary replications are made.

Local popularity

Each node uses the local request history of the corresponding content to decide if it is likely to be requested again in the future. If the rank of the content is among the best 30% the corresponding unit is replicated, so popular units are more likely to be replicated, but popularity information from neighbors is ignored. With this method the communication effort is minimized.

Neighbor popularity ranking

After collecting the popularity ranks for the content from the neighbors, the peer decides if it is worth to replicate the corresponding unit. The ranks are aggregated to a region rank (see (Sobe et al. 2009)), which is calculated as follows:

$$R = \frac{1}{n}\sum_{i=1}^{n} \ln(r_i)$$

n represents the number of neighbors and r is the rank of the specific unit at a neighbor i, where 0 is the best rank. To reduce the impact of peak ranks (e.g., one unit is best ranked at two nodes, but worst ranked on the third node) the logarithm is used. If the region rank R is lower than a given threshold (e.g., the best 30% at all neighbors) the unit is replicated.

Neighbor hormone ranking

Analogue to the popularity ranking the units can also be ranked by their hormone values at the neighbors. The higher the hormone value for a unit on a neighbor, the better is the unit's rank. The collected ranks can be aggregated as before and if the region rank is lower than a given threshold (e.g., the best 30%), the unit is replicated.

Clean-up/Replacement strategies

We concluded in (Sobe et al. 2011a) that efficient replacement or clean-up has to be done in order to avoid blocking the transport of units. A clean-up is triggered if a certain storage level is reached, which leads to a balanced storage load of the system.

However, if we want to have at least one instance of the unit available in the system, the choice of which unit to delete becomes more complicated. We solve this issue by applying a simple mechanism. Before deleting a unit, the node checks if at least one copy of it is on one of its neighbor nodes. We assume that the nodes follow a simple coordination protocol to avoid concurrent deletion of the last two units in a system.

The goal is to find an efficient strategy that does not increase the delay, but increases the utilization of replicas. We compare three mechanisms: least recently used (LRU), least frequently used (LFU) and hormone-based clean-up. LRU takes care of popularity changes of units, thus removes those units for which the most time steps passed since their last presentation. LFU targets units of low popularity, i.e., the least frequently presented units are removed. Hormone clean-up exploits local knowledge about hormone concentration.

A unit is deleted if there are no hormones for it on the neighbors, thus there is no current demand for it. Units currently in delivery are not deleted.

Simulation setup

We implemented a simulator[a]. At start-up, the simulator generates the network topology, initializes the nodes with initial storage, configures the client settings.

A simulation run is a loop of actions performed at each time step. First, the clients generate new requests or consume content presentations, after that the peers are diffusing hormones and moving content. For our evaluations we average the results of 10 simulation runs. One simulation run lasts for 500s, which is sufficient to show that the content placement stabilizes.

Our simulation evaluates the capability of nodes to adapt to a high number of new content units, therefore we randomly create content at the beginning of the simulation. Clients request content and consume content (watching units). To stress the system, clients can submit a new request immediately after the consumption of the former content.

We consider two scenarios. One where we assume a social event with 50 attendees (e.g., a fair for selected consumer groups), and a larger social event (e.g., a local triathlon) with 1,000 attendees. We assume that each person is represented by one node. For both scenarios different settings apply, which we describe in the following.

Network topology

We assume for small overlay networks of 50 nodes a connected Erdős-Rényi random graph with a diameter of 6. For larger networks, e.g., with 1,000 nodes, we assume a scale-free network topology. To generate such a network the Eppstein Power Law Algorithm (Eppstein and Wang 2002) is used. The algorithm gets as input a random graph and by repetitively removing and adding edges a power law distribution is reached. The network diameter of the scale-free graph is 13. The bandwidth was set to 100 Mbit/s. In a different scenario, we considered a lower bandwidth with comparable delay results (Szkaliczki et al. 2012).

Initial storage

At the beginning of a simulation run each node creates units until 30% (50 nodes) respectively 3% (1,000 nodes) of its storage of 900 MB is filled. This results in approx. 5,000 units for the 50 nodes scenario and 15,000 units for the 1,000 peers scenario. The units do not have any specific placement.

We expect that in a scenario with 50 motivated persons, each person will contribute with equal probability. In a scenario with 1,000 visitors we expect that there will be few highly motivated persons and a high number of less motivated persons. Thus, nodes with a higher degree will provide more storage space than others.

The average size of a unit is 2.6 MB, the maximum size is 16 MB and the minimum size is 190 KB, with a playback bit rate of 1 Mbit/s. These unit sizes are the result of a project use case “The long night of research”[b]. This use case was part of a university-local event where all research groups presented their work to the public, similar to a fair. We encouraged the visitors to upload their photos and videos to one of our servers. The visitors could browse and tag the uploaded content on a web-page, but were not able to share their content directly with other visitors. We analyzed the keywords for the uploaded content and additionally argue that one can compare the tag popularity to content popularity of user-generated content (Cha et al. 2007). Thus, we assume that the tags follow a Zipf-like distribution. On content creation, keywords are mapped to units, whereas one keyword might be used for several units.

Request generation

One request consists of 1–10 units identified by keywords. Thus, a client is requesting content types, not specific files. The request is fulfilled if for each keyword at least one unit is stored on the node. We further implemented a taste change, i.e., if a user likes the content just watched, her taste for future requests is with 10% probability similar to the currently watched unit.

In this paper we do not consider any order of the units, thus, if a requested unit arrives, it is presented to the user. Sequential and parallel dependencies have been handled for the delivery in (Sobe et al. 2010).

We further introduce a deadline for each unit, until which it has to be delivered. The deadline is dependent on the size, the link bandwidth and the maximum number of hops a unit can travel `maxhops` (see Equation 1).

$$deadline_u = t + maxhops \cdot \frac{size_u}{datarate} \tag{1}$$

If a deadline is missed, no further hormones for that unit are created to stop attracting content. A request is considered as failed if none of the requested units could fulfill their deadline.

A user can only submit one request at a time. If this request is fulfilled or failed, a new request will be generated. This leads to around 10,000 requests during a simulation run in the 50 nodes scenario and around 400,000 requests for the 1,000 nodes scenario. A failed request usually leads faster to the creation of a new request than the presentation of units, thus the number of requests sent during a simulation might differ. Therefore, the metrics will consider the ratio of successful and submitted requests.

Simulation parameters

We used the evolutionary algorithm as described before with a fitness function targeting client satisfaction by optimizing the number of successful requests, ($f = \frac{requests_{fulfilled}}{requests_{submitted}}$). The evolutionary algorithm is part of our open source simulator that also runs the artificial hormone-system. For evaluating the fitness of a parameter set, the simulation is started with this parameter set for a number of runs and the results are averaged (we used hormone ranking replication with hormone clean-up). The parameter sets of one population are compared according to their fitness and the result of one generation is the parameter set with the highest fitness. The higher the number of generations the higher is the fitness of the resulting parameter set. The resulting parameter set can be used for all simulations and real implementations of the algorithm for which the system's configuration (e.g., number of nodes, replication type, etc.) is similar to the input of the genetic algorithm.

In Table 2, the resulting parameters are shown. The hormone creation values for a unit, η_0 and η, are high and hence lead to a wider travel range. Additionally, the diffusion of 45% (α) of the hormones supports longer travel distances. Compared to the creation amount η,

Table 2 Parameter settings

η_0	3.95
η	4.39
α	0.45
ϵ	0.16
t	0.23
m	0.23
c	60%
maxhops	10

Parameter configuration, optimized by a genetic algorithm.

the evaporation rate ϵ is low, meaning the hormones last for some time. The evaporation of hormones is a fixed value subtracted from the current hormone level. The migration threshold m describes the minimum hormone difference between two nodes to make a unit move. In this case the difference is very low in comparison to the creation amount of hormones. This leads to a very dynamic behavior of the units. The clean-up (c) is triggered by a node if its current storage level exceeds 60% (50 nodes) resp. 30% (1,000 nodes).

Metrics

We aim at evaluating both the request fulfillment and the utilization of replicas. The fulfillment of a request is quantified by the *delay*. The delay of a unit is shown in equation 2 is measured from the request time of a unit until the arrival of that unit on the node. A delay of 0 s means that the unit was already on the node.

$$\delta = t_{arrival} - t_{req} \tag{2}$$

For better readability the delay is presented as cumulative distribution function (CDF) (Gubner 2006). It shows the likelihood of a delay value to be less than or equal to a certain value at a given point during the simulation.

The *deadline miss rate* represents the number of units out of all units U (not requests), for which the deadline is missed (see Equation 3).

$$M := \{u_i | u_i \in U : \delta_{u_i} > t_{deadline} - t_{req}\}$$

$$\text{deadline miss rate} = \frac{|M|}{|U|} \tag{3}$$

If a unit missed its deadline, the delay is calculated as deadline minus request time (max. delay) as shown in equation 4.

$$\delta_{max} = t_{deadline} - t_{req} \tag{4}$$

The *request failed rate* indicates requests R_i out of submitted requests R, from which all units u_i missed their deadline as shown in equation 5.

$$F := \{u_i | u_i \in R_i, R_i \subseteq R : \delta_{u_i} \geq \delta_{max}\}$$

$$\text{request failed rate} = \frac{|F|}{|R|} \tag{5}$$

A unit is presented for some time, and as *unit utilization* we measure the rate of units (u_i) that are currently being presented (are part of P). The more unit presentations started in comparison to the number of their replicas U_i, the better the *unit utilization*.

$$P := \{u_i | u_i \text{ currently being presented}\}$$
$$C_i := \{r_i | \exists u_i \in P : r_i \text{ is replica of } u_i\}$$

$$\text{utilization} = \sum_{i=0}^{|P|} \frac{1}{|C_i|} \tag{6}$$

The utilization and the request failed rate will be depicted as box plot with 1.5 interquartile range whiskers.

Results and discussion

As explained in the former section we apply two scenarios, a 50 nodes scenario and a 1,000 nodes scenario. We investigate combinations of replication mechanisms and clean-up mechanisms and their impact on delay, utilization and request failure rate. We further analyze the impact of node failure to the replication and clean-up mechanisms.

In the following we list the used replication techniques and their names used in the figures alphabetically:

- `hormone` replicates if for the given unit hormones exist on the neighbors
- `hranking` ranks the hormones of neighbors to decide the need of replication
- `owner` only replicates on the requesting node
- `path` replicates always
- `path_adapt` replicates with a given probability on the current node
- `pop` replicates if the popularity of this unit is high enough on the current node
- `pranking` ranks the unit's popularity in the neighborhood to decide the need of replication

50 Nodes random network

In this scenario we start with a wrap-up of the base replication scenarios without any clean-up, then we show the differences, if clean-up is applied.

In Figure 2 the base replication cases are shown as CDF. It depicts the likelihood of delays to be below or equal a specific value. Thus, the more the curve is located on the left side, the lower is the delay. In the legend we order the replication names according to their resulting delay. `owner` replication leads to the highest delay, followed by `hormone` and `pranking`, then `path_adapt` and `pop`. It can be seen that `hranking` even outperforms the `path` replication mechanism. The reason for this is that `path` replication creates too many copies and therefore some nodes do not have any storage space left for transport[c]. The storage space balancing is not only a problem of `path` replication, the other replication mechanisms also fill a number of nodes that cause the blocking of transport. In the following we compare the delay, the request failure rate and the utilization of the three introduced clean-up mechanisms hormone, LRU and LFU.

In Figure 3 the delay CDF for hormone clean-up is shown. In general, the order of delay curves remains the same except for `path` and `hranking`, which switch their position.

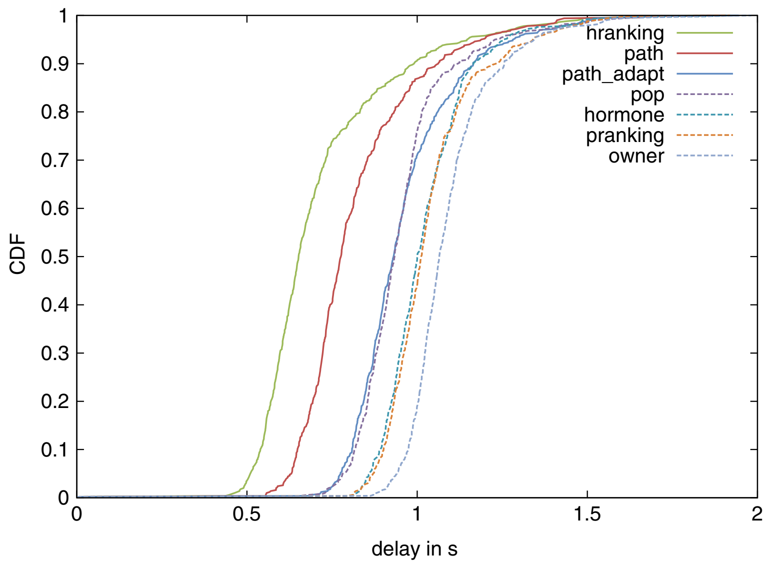

Figure 2 No clean-up. Delay of replication mechanisms without clean-up in a 50 nodes random network.

However, the clean-up has a negative impact on all replication mechanisms as their delay curves are located closer to 1s. In comparison to the base case, the most conspicuous example is the combination with local popularity replication (pop), where a high delay jitter results in a gentle curve. The following hormone delay curve shows a similar, although less extreme delay jitter increase.

The LRU clean-up is depicted in Figure 4. With this clean-up mechanism path replication is also affected, which results in a similar delay of path, hranking and path_adapt. LRU further increases the delay jitter of path as shown by a more gentle

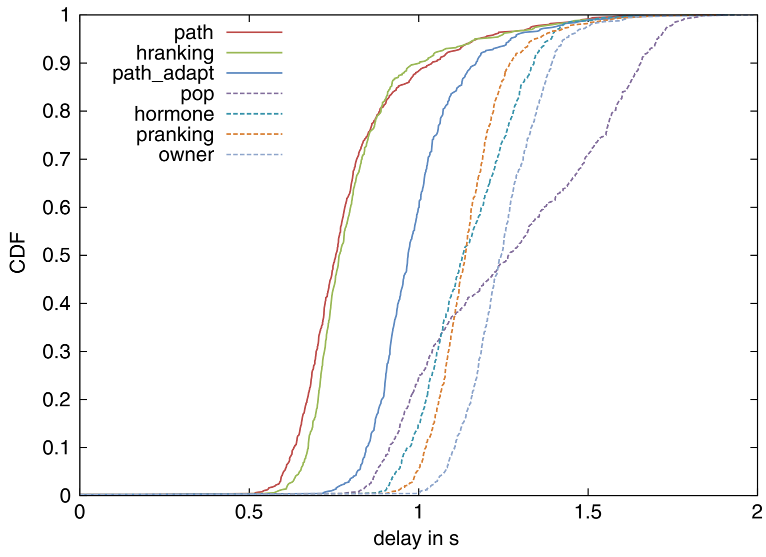

Figure 3 Hormone clean-up. Delay of replication mechanisms in combination with hormone clean-up in a 50 nodes random network.

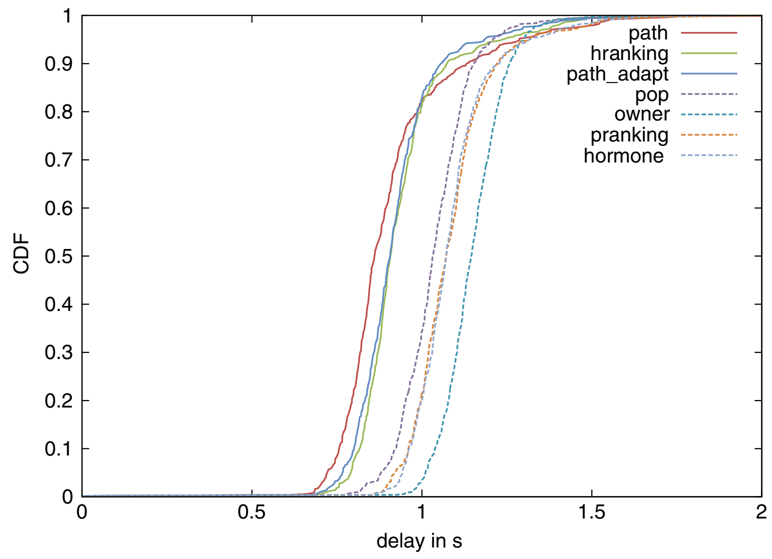

Figure 4 LRU clean-up. Delay of replication mechanisms in combination with LRU clean-up in a 50 nodes random network.

curve than `hranking`. In this graph the delay of `hormone` is even higher than `owner` replication.

LFU, as depicted in Figure 5, results in high delay increases for all replication mechanisms. In particular the `hormone` replication mechanism and `owner` replication suffer from wrong decisions and result in gentle curves. `path` replication results in doubled delay, which leads to a similar delay as `pop`. `hranking` results in a similar delay as `path_adapt`, but with a lower delay jitter its curve is steeper.

When comparing the different clean-up mechanisms, one can see that all of them have a negative impact on the delay. In general, hormone clean-up has the lowest impact if

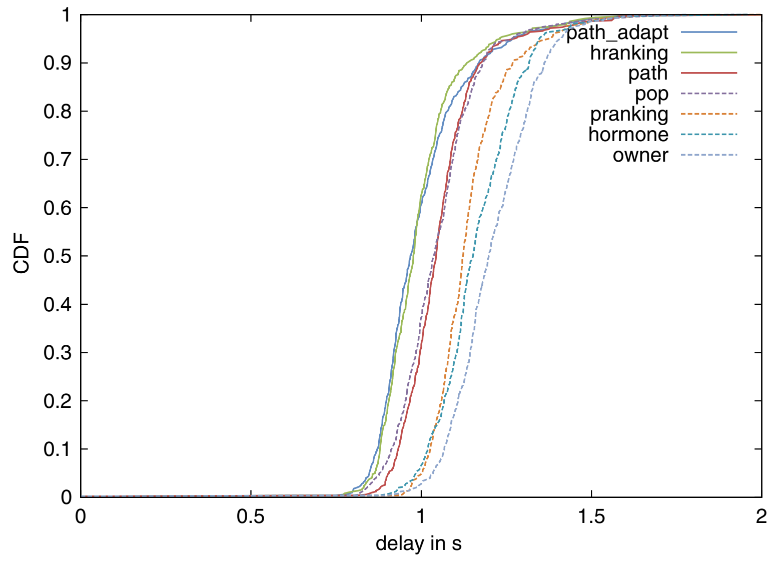

Figure 5 LFU clean-up. Delay of replication mechanisms in combination with LFU clean-up in a 50 nodes random network.

considering the delay of the best combination (here, path replication). However, it has a very bad impact on pop and hormone replication and further results in the highest delay for owner. LRU has the lowest impact on path_adapt in comparison to the base case. Although the delay of the replication mechanisms increases it leads to steeper curves than hormone clean-up and LFU. LFU has the highest impact if considering the delay increase of the best combination (here, path_adapt replication).

hormone replication seems to suffer from any clean-up. The reason is the low number of replicas created by this mechanism. Additionally, the placement is not ideal such that any removal is crucial. pranking replication does not create enough replicas and therefore performs similar to hormone replication. The combination of pop and hormone clean-up does not fit, because the replication mechanism considers long-term popularity (based on the history of the requests) and the clean-up mechanism considers short-term popularity. hranking replication creates many replicas on strategic positions and therefore the transport might be blocked.

Although the clean-up mechanisms have a negative impact on the delay, it is interesting to see whether the utilization of the units has been improved.

The utilization measures the improvement in storage efficiency of the single replication mechanisms. In Figure 6 we show the box plot of all combinations, including the base case. We group the results per replication mechanism.

In comparison to the base case, hormone replication does not take advantage of any clean-up mechanism, because it already creates only a small amount of replicas. The utilization of path and pop replication lowers if clean-up is applied. path_adapt and owner lead in any combination to a stable utilization. The highest utilization improvement, in comparison to the base case, is shown by hranking in combination with hormone clean-up. Also LFU would make a good fit, but results in a higher delay.

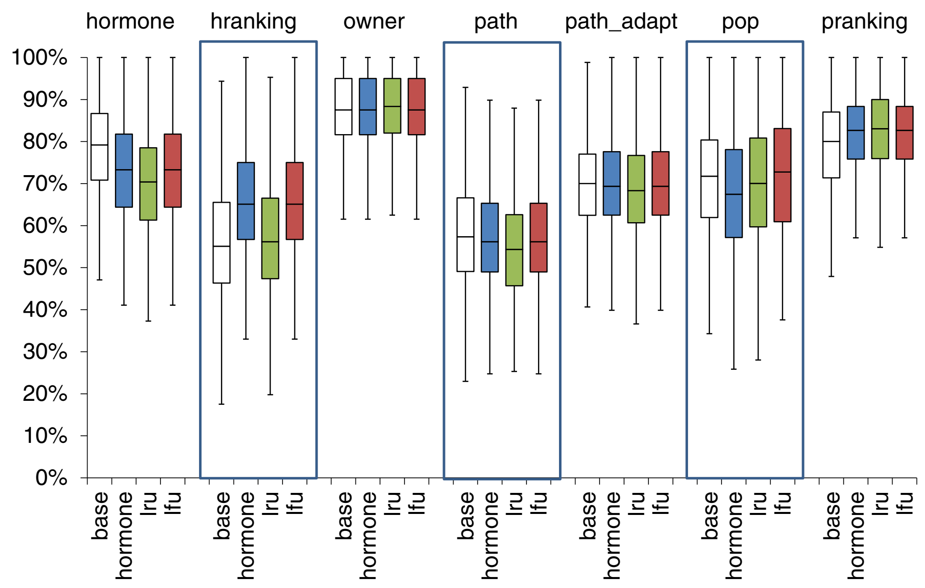

Figure 6 Utilization. Utilization comparison of hormone clean-up, LRU and LFU in a 50 nodes random network.

path replication leads to the lowest utilization if combined with LRU. As already seen in the delay comparison, pop replication with hormone clean-up do not fit. hranking has the worst utilization if combined with LRU. The highest utilization is reached by owner replication, because it creates the lowest number of replicas. However, has also the highest delay.

The utilization shows the effects of the clean-up, however the quality of the approaches are further evaluated by the capability to fulfill user requests. We compare the failed request rate in the following paragraphs.

Figure 7 groups the box plots of each replication and clean-up combination including the base case. One can see that the average failed rate of all combinations is around 10 to 20%. The lowest failed rate is reached by hranking and path replication. The highest failed rate is reached by owner replication. This indicates that replicas on the transport path improve the service quality. As already shown by the delay and the utilization, pop replication with hormone clean-up does not fit. The outliers marked by (*) can be explained by high failure rates at the beginning (within the first 100s) of the simulation. owner and hormone replication show the highest variance of failure rates.

In general, the failure rate increases in comparison to the base case, because the number of replicas is reduced and not all future requests can be predicted. The lower the increase of the failure rate the better the combination fits. However, owner replication with LRU, path_adapt with LRU and path with hormone clean-up manage to lower the failure rate in comparison to the base case. This can be explained by a better balancing of the placement. LRU leads to the lowest increase of the failure rate with most of the approaches, except the combination of hranking and path with hormone clean-up. LFU leads to the highest increase of the failure rate.

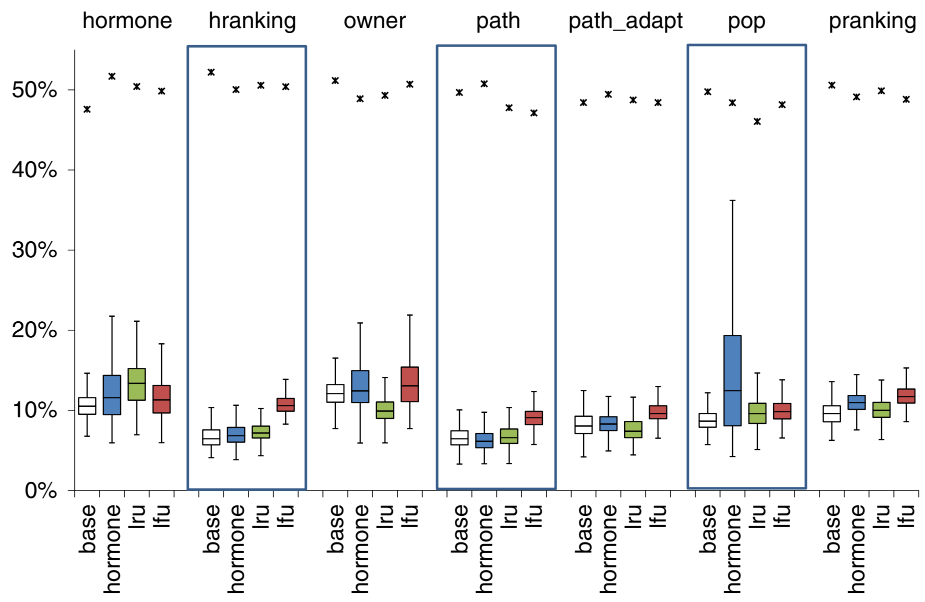

Figure 7 Failed requests. Hormone, LRU and LFU request failed rate comparison in a 50 nodes random network.

Considering delay, utilization and failure rate we select two of the best approaches and continue our discussion only with them. Although `path` replication has the lowest delay it creates too many replicas and leads to the lowest utilization and therefore we do not further consider this approach. We further omit all cases with LFU clean-up, because it leads to the highest delay, the lowest utilization and the highest failure rates.

There is no replication mechanism that is best in all cases. The second best candidates regarding delay are `hranking` and `path_adapt`. `hranking` has the lowest delay in combination with hormone clean-up. This combination leads to the best improvement of utilization in comparison to the base case and only results in a slight increase of the failure rate. The lowest impact on the delay has the combination of `path_adapt` and LRU, which is further leading to a lower failure rate, however also a slightly lower utilization as the combination with hormone clean-up. We decide for combining `path_adapt` with LRU because of higher chances to fulfill user requests.

A discussion of these two approaches is further interesting, because `hranking` is based on neighborhood knowledge and `path_adapt` represents an uninformed (traditional) approach.

Impact of node failure

We simulate node failure as randomly chosen nodes being removed periodically one-by-one. We do not handle isolated nodes after peer deletion. Thus, there is a performance gain potential if an overlay algorithm takes care of reconnecting such nodes. In other scenarios we also add new nodes (Szkaliczki et al. 2012), however, the results differ only marginally. We chose 5, 10 and 20 nodes to be removed. If nodes fail it cannot be guaranteed anymore that one copy of a unit exists, however, since a request consists of keywords and not unit ids, it is enough to have at least one unit per keyword. The scenario should further show how good the algorithms adapt the unit placement and change the transport paths.

Figure 8 shows the delay distribution of the `hranking` algorithm with hormone clean-up. One can see that it is capable of handling loss. Overall, the delay increases slightly, but interestingly the delay of 5, 10 and 20 removed nodes is very similar. This shows that the transport paths adapt fast to the new conditions.

The `path_adapt` replication with LRU leads to interesting results as shown in Figure 9. Here, the delay of the peer failure scenarios is lower than in the original case. This anomaly can be explained by referencing the clean-up failures of the original scenario. A clean-up fails if on the current node all units are currently in use or there is no copy of the current unit on a neighbor. A disadvantageous replica distribution might be that at every second hop a replica is placed, which means that a high number of replicas exist in the system, but because the nodes only see their neighbors, the units cannot be deleted. In the peer failure scenario, it is likely that a node that blocks the transport is removed and therefore opens the possibility of alternative paths. Hence, to reduce the delay of the original case, a clean-up mechanism would have to check the neighborhood with a larger hop-distance.

In Figure 10 we compare the request failure rates in case of node failure. One can see that the failure rates only increase slightly, which shows that both `path_adapt` and

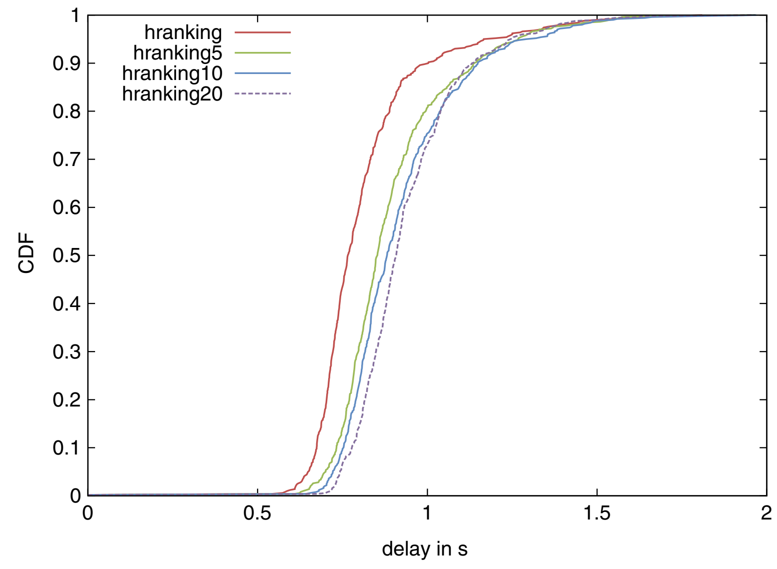

Figure 8 Node failure - hormone ranking. Delay distribution of hormone ranking with hormone clean-up if 5, 10, 20 nodes fail.

hranking are robust. However, if the number of failed nodes gets higher than 20, we expect a performance drop.

Both approaches are performing similarly for a small network, however, with a lower delay of hranking, but a more stable failure rate is reached by path_adapt if nodes fail. hranking places the units more efficiently, its utilization can be increased by a clean-up mechanism.

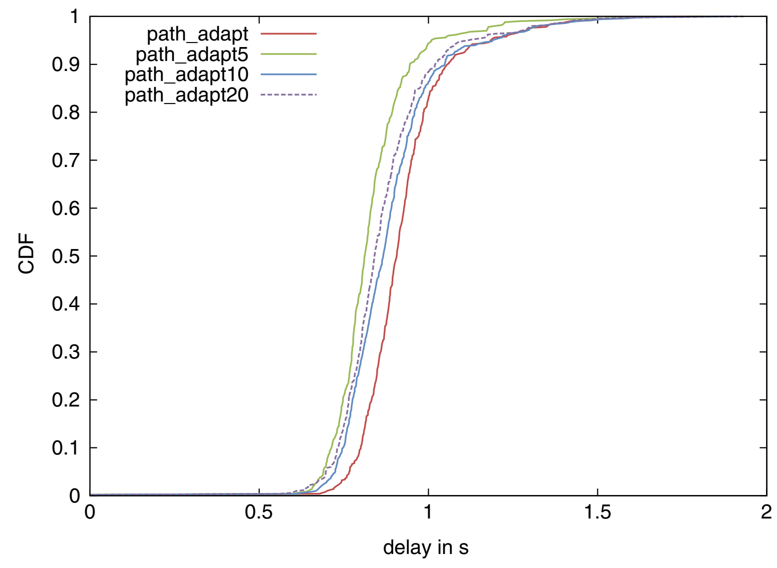

Figure 9 Node failure - path adaptive. Delay distribution of path adaptive with LRU if 5, 10, 20 nodes fail.

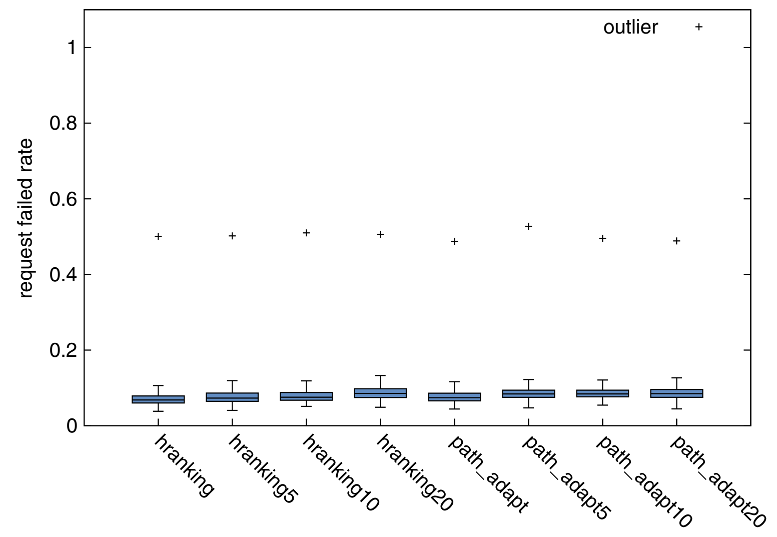

Figure 10 Failed requests and node failure. Failed request rate comparison in case of node failures in a 50 nodes network.

1,000 nodes scale-free network

In this section we evaluate the applicability of our delivery algorithm for scale-free networks. It is shown that the parameters for the 50 node network also work for the 1,000 node network. Specific optimization using the genetic algorithm could lead to even better results. For the delay comparison we immediately include the delay of failed nodes.

Figure 11 depicts the delay of `hranking` with hormone clean-up. In comparison to the small network scenario the delay is increased by around 500 ms and the curve is less steep. The paths in a scale-free network scenario are not as long as

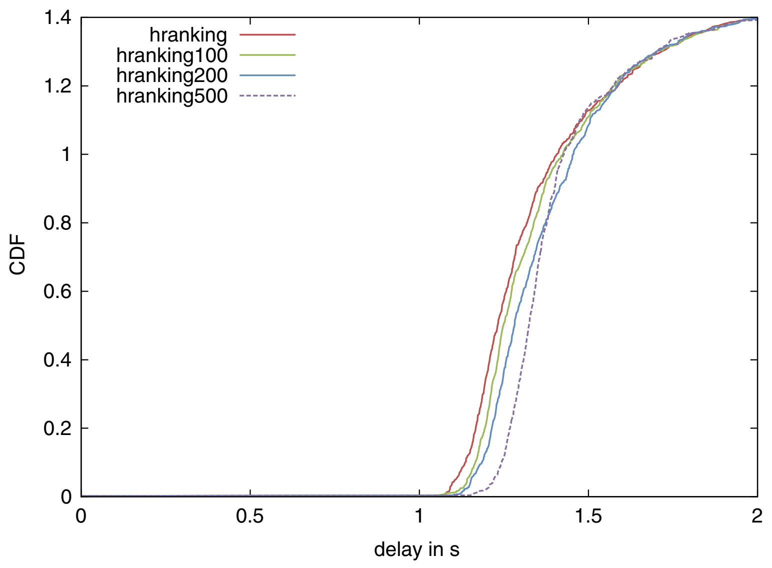

Figure 11 1000 nodes - hormone ranking. Delay distribution of hormone ranking with hormone clean-up if 100, 200, 500 nodes fail.

in a random network. The best performance is reached if the units are placed on high-degree nodes.

If 100, 200 and even 500 nodes fail the delay does not increase considerably, even less than in the small network scenario. Note that also high degree nodes may fail, because the nodes leaving the network are chosen randomly. Thus, only replicating on high-degree nodes would cause service interruptions in case of node failures. The replication and clean-up mechanisms find the right balance of placement, which can be explained by low delay increases.

Figure 12 shows that for `path_adapt` the problem with clean-up failures does not apply such as in the 50 nodes network. The reason is the network structure and its rather low diameter. Therefore, the delay of `path_adapt` replication is only a bit higher to the delay of `hranking`.

Both algorithms show slight increases of request failures also in the presence of node failure (see Figure 13). Overall, the request failed rate is 20% in comparison to around 10% in the small network case. Also the variance of the values is higher. The `hranking` replication is more robust than `path_adapt`.

Discussion

Although the results of the replication mechanisms are promising, the clean-up has a negative impact on the delay, which means that the goals of the clean-up are not reached. Ideally, a combination of replication and clean-up should lead to low delay, low failed rate and high utilization. If the settings regarding unit deletion are as strict as in this paper, `path` replication with hormone clean-up, although inefficient, performs best regarding delay. Alternatives could be `path_adapt` replication with LRU and `hranking` replication with hormone clean-up. The node failure scenarios showed that there are still nodes, which block the transport of units. To solve this issue the settings could be less strict regarding the deletion policy. Instead of deleting a unit only if there is a copy of it in the

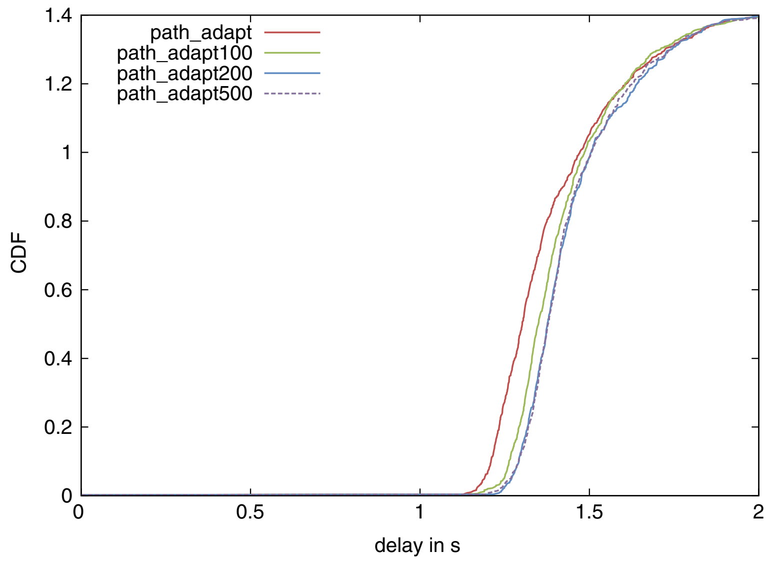

Figure 12 1000 nodes - path adaptive. Delay distribution of path adaptive with LRU if 100, 200, 500 nodes fail.

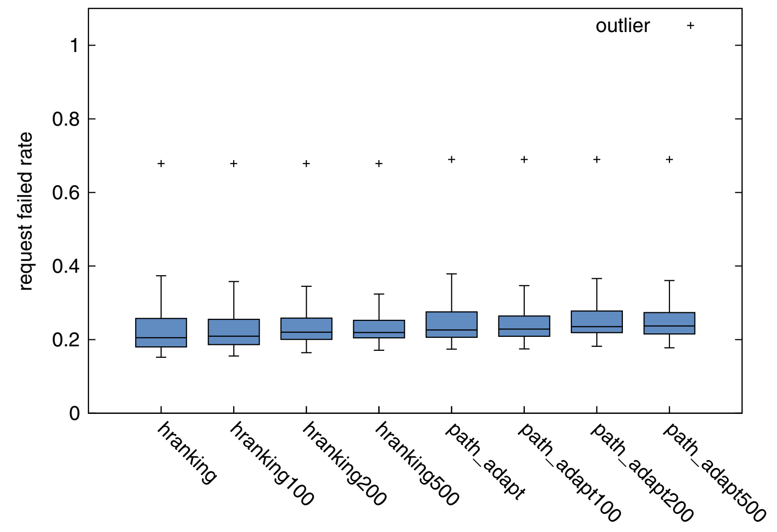

Figure 13 1000 nodes - failed requests. Failed request rate comparison in case of node failures in a 1000 nodes network.

neighborhood, it could be weakened to delete a unit if another unit covering the same keyword is in the neighborhood.

Conclusion

Replication is a typical measure to increase robustness, but usually nodes in a network do not provide unlimited storage space. A clean-up or replacement mechanism is necessary to balance the available storage space. If a network of nodes is fully connected, and a central server knows how many copies exist in the system, the decision to delete one of the copies is easy (although it is not easy to choose the optimal number of replicas in the system). In self-organized networked systems, where the knowledge of a node is limited to its neighborhood, storage balancing is a challenge. In this paper we investigated different replication mechanisms in combination with three clean-up mechanisms, under the constraint that content must not vanish from the system. The basis is a hormone-based delivery algorithm that allows for quality-aware content distribution based on local knowledge. We evaluated the delay impact of the clean-up mechanisms, as well as the impact on the utilization of replicas. Although reducing the number of replicas, the robustness of the system is still given. Similar results are reached if the system is applied to a scale-free network of 1,000 nodes.

The evaluated clean-up mechanisms are either uninformed, such as least recently used (LRU) and least frequently used (LFU), or use the hormone information of the neighborhood. We showed that it is necessary to match the replication mechanisms with the clean-up mechanisms, otherwise a major delay drop can be experienced. We also found out that uninformed replication mechanisms like path adaptive replication (random replication on the delivery path) in combination with LRU might be an alternative for dynamic delivery that does not rely on any indicator of interest such as hormones or pheromones. The hormone-based clean-up mechanism might be mapped to pheromone-based delivery systems, where specific ants could check the neighborhood for interest in a given content unit. As we have seen that any of the clean-up mechanisms influences the delay, future work should regard the replication mechanisms, i.e., to avoid unnecessary replicas in the system. Furthermore, an alternative deletion policy might be introduced that is less restrictive than the one chosen in this paper.

Endnotes

[a] The simulator used in this section is published under GNU GPL at http://code.google.com/p/videonetwork/

[b] SOMA Web-page http://soma.lakeside-labs.com/?p=253

[c] For a more detailed discussion of the base case please refer to (Sobe et al. 2011a)

Competing interests
Both authors declare that they have no competing interests.

Authors' contributions
AS conducted the simulation scenarios, implemented the artificial hormone algorithm and drafted the manuscript. WE implemented the genetic algorithm, helped with the concept of the artificial algorithm and contributed with text to the manuscript. All authors read and approved the final manuscript.

Acknowledgements
This work was supported by the Lakeside Labs GmbH, Klagenfurt, Austria, and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214|21532|32604 (MESON), and KWF 20214|22935|34445 (SMART MICROGRID). We would like to thank Lizzie Dawes for proofreading the paper.

Author details
[1]Institute of Networked and Embedded Systems/Lakeside Labs, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria. [2]Computer Science Department, University of Neuchâtel, Neuchâtel, Switzerland. [3]Complex Systems Engineering, Universität Passau, Passau, Germany.

Received: 28 September 2012 Accepted: 22 April 2013
Published: 24 June 2013

References

Androutsellis-Theotokis S, Spinellis D: **A survey of peer-to-peer content distribution technologies.** *ACM Comput Surv* 2004, **36**(4):335–371.

Böszörmenyi L, del Fabro M, Kogler M, Lux M, Marques O, Sobe A: **Innovative directions in self-organized distributed multimedia systems.** *Multimedia Tools Appl, Springer* 2011, **51**(2):525–553.

Brinkschulte U, Pacher M, Renteln A: **Towards an artificial hormone system for self-organizing real-time task allocation.** In *Software Technologies for Embedded and Ubiquitous Systems, Volume 4761 of Lecture Notes in Computer Science*. Edited by Obermaisser, R, Nah Y, Puschner P, Rammig F. Berlin Heidelberg: Springer; 2007:339–347.

Cha M, Kwak H, Rodriguez P, Ahn Y, Moon S: **I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system.** In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. San Diego: ACM; 2007:1–14.

Clarke I, Sandberg O, Wiley B, Hong T: **Freenet: A distributed anonymous information storage and retrieval system.** In *Designing Privacy Enhancing Technologies Volume 2009 of Lecture Notes in Computer Science*. Edited by Federrath H. Berlin Heidelberg: Springer; 2001:46–66.

Cohen B: **BitTorrent Protocol Specification.** 2003. [http://www.bittorrent.org/beps/bep_0003.html]

Cohen E, Shenker S: **Replication strategies in unstructured peer-to-peer networks.** In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '02. New York: ACM; 2002:177–190.

Dorigo M, Birattari M, Stutzle T: **Ant colony optimization.** *IEEE Comput Int Mag* 2006, **1**(4):28–39.

Elmenreich W, Klingler G: **Genetic evolution of a neural network for the autonomous control of a four-wheeled robot.** In *Sixth Mexican International Conference on Artificial Intelligence, Special Session*. Aguascalientes: IEEE; 2007:396–406.

Eppstein D, Wang J: **A steady state model for graph power laws.** In *International Workshop on Web Dynamics, WebDyn 2002*. Honolulu: arxiv e-print, arxiv:cs/0204001; 2002.

Forestiero, A, Mastroianni C, Spezzano G: **Building a peer-to-peer information system in grids via self-organizing agents.** *2007 2nd Bio-Inspired Models Netw, Inf Comput Syst* 2007, **6**(2):125–140.

Forestiero, A, Mastroianni, C, Spezzano, G: **QoS-based dissemination of content in grids.** *Future Gen Comput Syst* 2008, **24**(3):235–244.

Gubner, J: *Probability and Random Processes for Electrical and Computer Engineers*. New York: Cambridge University Press; 2006.

Hausheer, D, Nikander P, Fogliati V, Wünstel K, Callejo M, Jorba S, Spirou S, Ladid L, Kleinwächter W, Stiller B, Behrmann M, Boniface M, Courcoubetis C, Man-Sze L: **Future internet socio-economics - challenges and perspectives.** In *Towards the Future Internet*. Netherlands: IOS Press Amsterdam; 2009:1–11.

Herrmann K: **Self-organizing replica placement-a case study on emergence.** In *Self-Adaptive and Self-Organizing Systems, 2007. SASO'07*. Cambridge: IEEE; 2007:13–22.

Kulis B, Grauman K: **Kernelized locality-sensitive hashing for scalable image search.** In *IEEE 12th International Conference on Computer Vision, ICCV2009*. Kyoto: IEEE; 2009:2130–2137.

Lee S B, Muntean GM, Smeaton AF: **User-centric utility-based data replication in heterogeneous networks.** In *IEEE International Conference on Communications Workshops, ICC Workshops 2008*. Beijing: IEEE; 2008a:290–294.

Lee S B, Muntean GM, Smeaton AF: **Smart PIN: Utility-based replication and delivery of multimedia content to mobile users in wireless networks.** In *2008 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. Las Vegas: IEEE; 2008b:1–8.

Leontiadis E, Dimakopoulos V, Pitoura E: **Creating and maintaining replicas in unstructured peer-to-peer systems.** In *Euro-Par 2006 Parallel Processing Volume 4128 of Lecture Notes in Computer Science*. Edited by Nagel W, Walter W, Lehner W. Berlin Heidelberg: Springer; 2006:1015–1025.

Lv Q, Cao P, Cohen E, Li K, Shenker S: **Search and replication in unstructured peer-to-peer networks.** In *Proceedings of the 16th International Conference on Supercomputing*. New York: ACM; 2002:84–95.

Mamei M, Menezes R, Tolksdorf R, Zambonelli F: **Case studies for self-organization in computer science.** *J Syst Arch* 2006, **52**(8–9):443–460.

Michlmayr E: *Ant algorithms for self-organization in social networks*. Austria: Technical University Vienna; 2007.

Rong L: **Multimedia resource replication strategy for a pervasive peer-to-peer environment.** *J Comput* 2008, **3**(4):9–15.

Sobe A: *Self-organizing multimedia delivery*. Klagenfurt: Alpen-Adria Universität; 2012.

Sobe A, Böszörmenyi L: **Non-sequential Multimedia Caching.** In *2009 First International Conference on Advances in Multimedia (MMedia 2009)*. Colmar: IARIA/IEEE; 2009:158–161.

Sobe A, Elmenreich W, Böszörmenyi L: **Towards a self-organizing replication model for non-sequential media access.** In *ACM MM Workshop on Social, Adaptive and Personalized Multimedia Interaction and Access (SAPMIA)*. Florence: ACM; 2010:3–8.

Sobe A, Elmenreich W, Böszörmenyi L: **Replication for bio-inspired delivery in unstructured peer-to-peer networks.** In *Ninth Workshop on Intellingent Solutions for Embedded Systems (WISES 2011)*. Regensburg: IEEE; 2011a:6.

Sobe A, Elmenreich W, Böszörmenyi L: **Storage balancing in self-organizing multimedia delivery systems.** *Technical Reports, Institute of Information Technology, Alpen-Adria Universität Klagenfurt, Austria, arxiv e-print, arxiv:1111.0242* 2011b, **TR/ITEC/01/2.13:**16.

Szkaliczki T, Sobe A, Böszörmenyi L: **Discovering bounds of performance of self-organizing content delivery systems.** In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE Sixth International Conference on*; 2012:97-104. doi:10.1109/SASOW.2012.26.

Xu Qz, Wang L: **Recent advances in the artificial endocrine system.** *J Zhejiang Univ SCI C* 2011, **12**(3):171–183.

Yamamoto H, Maruta D, Oie Y: **Replication methods for load balancing on distributed storages in P2P networks.** In *The 2005 Symposium on Applications and the Internet, SAINT 2005*. Trento: IEEE; 2005:264–271.

doi:10.1186/2194-3206-1-13
Cite this article as: Sobe and Elmenreich: Replication and replacement in dynamic delivery networks. *Complex Adaptive Systems Modeling* 2013 **1**:13.