SEAHORSE: Generalizing an artificial hormone system algorithm to a middleware for search and delivery of information units

Anita Sobe[1,*], Wilfried Elmenreich[2], Tibor Szkaliczki[3] Laszlo Böszörmenyi[4]

[1] *Computer Science Department, University of Neuchatel, Switzerland*
[2] *Institute of Networked and Embedded Systems, Alpen-Adria Universität Klagenfurt, Austria*
[3] *E-learning department, MTA SZTAKI, Hungary*
[4] *Institute of Information Technology, Alpen-Adria Universität Klagenfurt, Austria*
∗ *E-mail: anita.sobe@unine.ch*

Abstract

This paper introduces SEAHORSE (SElforganizing Artificial HORmone SystEm), a middleware that builds upon an artificial hormone system for search and delivery of information units. SEAHORSE is a generalization of an artificial hormone algorithm where information units are requested by network nodes via emitting a an artificial hormone which is propagated through the network with respect to the current network conditions. Information units are following the hormone gradient and therefore place themselves on servers where they are close to the requesting nodes. This self-organizing algorithm is robust and scalable, however, due to their complex nature, self-organizing systems are hard to configure and set up to get a desired outcome. Parameter settings that work on different scales are crucial for making the system work. Therefore, we provide a parameter study based on two use cases showing the applicability of SEAHORSE to target applications ranging from multimedia distribution at social events to information dissemination in smart electrical microgrids.

Keywords: Dynamic Networks, Self-organization, Middleware

1. Introduction

With the rise of networked smart devices and the so called Internet of Things, services require more scalability and robustness to handle the complexity of the underlying ecosystem. In some situations (e.g., disaster areas, large-area sports events, battle fields, etc.), a traditional network infrastructure does not exist, or is expensive to set up. In this context content is also consumed in a more dynamic way than in traditional environments [1].

The provided system has to be scalable, robust and adaptive, which is usually hard or impossible to implement in a centralized manner. If looking at principles found in nature we can see that it is possible to handle complexity and dynamics by relying exclusively on simple, local decisions (bio-inspired self-organizing systems).

As an example, ants are exploring the surrounding area to find food, and if found they go back to their home base leaving pheromones to guide other ants. It has been proven that with such a simple mechanism the shortest path between nest and food source can be found even in complex environments [2]. The shortest path, however, is often not enough, especially when multimedia content has to be delivered. Quality of Service (QoS) bounds (delay, packet loss, etc.) are necessary to make the desired content consumable without artifacts and interruptions.

Another class of bio-inspired self-organizing algorithms are artificial hormone systems[3] which are based on the endocrine system adjusting the metabolism of tissue cells in our body [4]. These systems commonly try to remodel the endocrine system to understand the self-organization of single cells. As a basis, hormone-like systems are powerful for any network-related use case. However, there is no biologically inspired algorithm that fits all requirements and researchers tend to extend either existing models or construct their own.

As stated in [5], designing a self-organizing system (where it does not matter if it is bio-inspired or not) is a challenging task. A self-organizing system consists of interacting local entities, of which global behavior arises. Thus, a self-organizing system can be considered as black-box system, which is usually tested and analyzed within a simulated environment before implementing it as a real-world application. One of the things that is often neglected is the configuration complexity of such an application. Typically, parameters are interdependent and in many self-organizing algorithms the proper settings take on a crucial role, i.e., with bad parameter settings the system might perform badly or even not work at all.

In previous works [6], [7], [8], [9] we have designed an artificial hormone algorithm that combines search and delivery of multimedia content in complex and dynamic networks. We have shown that, if the initial parameters are set properly, the artificial hormone algorithm performs well in comparison with state-of-the art techniques.

In this paper, we show the application of such an algorithm within a bigger picture. Specifically, we show, how a bio-inspired, self-organizing algorithm can be generalized to a middleware, which we name SEAHORSE (SElforganizing Artificial HORmone SystEm). The middleware separates the application from the routing and network architecture. As these elements should be interchangeable, interfaces are required, which are also defined in this paper. Self-organizing algorithms tend to be complicated to configure for the given set up. Usually, simulations are used to help in this context. However, to make a middleware applicable in a real system, it is necessary to reduce the configuration complexity. Thus, first we reduce the number of parameters and we evaluate the quality of the remaining parameters with the help of a fitness function. To show the remaining effort for applying SEAHORSE, we use two case studies from different contexts, one in the context of multimedia distribution and the other in the context of information dissemination in smart energy networks.

In the first use case, SEAHORSE enables tens of thousands of participants of a social event (for example spectators at a sports event such as a triathlon) to produce and share multimedia content continuously and instantaneously. This is such a complex and dynamic problem that self-organization is the only adequate approach to address it. In addition to that, being able to transfer multimedia content in the proper quality shows that any other type of content can be handled easily.

While in the first case, larger content types are considered, we also show that content that needs to reach network coverage as fast as possible can also be handled. In the second use case SEAHORSE serves for exchanging power consumption data in a smart electrical power grid. Although, usage patterns and parameters differ for both use cases, they have in common a complex and highly dynamic communication pattern, which calls for self-organization. SEAHORSE can be applied to both cases easily.

The structure of this paper is as follows. In Section 2 we give an overview of related works on bio-inspired search and routing. In Section 3 we describe the middleware architecture including interfaces and artificial hormone algorithm. In Section 4 we show the applicability of the middleware within two case studies followed by a parameter analysis and performance comparison. Finally, we give an outlook on future work and conclude the paper.

2. Related Work

Our system relates to works on search and routing, in particular those which use bio-inspired algorithms [10]. There are many principles found in nature that are scalable, adaptive and robust [11] and researchers adapt this behavior to their engineering problems. While there exists a considerable amount of applications of bio-inspired mechanisms, the combination of bio-inspired mechanisms and content delivery and dissemination is an emerging topic. Moreover, there exist many bio-inspired routing problems, while we address a self-organizing content placement in this paper.

Artificial Hormone Systems have been applied in technical systems such as task allocation [12] or synthesis of robot controller software [13]. Trumler et al.[14] propose an Artifial Hormone System as middleware for ubiquitous computing, as for example in smart office applications. The PRoPHET protocol [15] addresses the routing problem for opportunistic networks by calculating predictability value for the forwarding preferences at each node and updating these values based on encounters between nodes in packet forwarding. Another prominent example for stigmergic bio-inspired computing

is *AntNet* [16], which is inspired by the behavior of ants searching for resources for solving complex problems such as the Traveling Salesmen Problem [17].

Voulgaris et al. describe Sub-2-Sub, a self-organizing content-based publish/subscribe system for dynamic collaborative networks [18].

Kolisch [19] approaches the dynamic replica placement problem with simulated annealing, but under the requirement of full knowledge of the network in order to perform the optimization. In contrast, our algorithm is using local information as a basis for decisions on migrations or replication of content.

In dynamic grid networks a major task is to discover services and resources. For faster lookup, descriptors are disseminated over the grid, i.e., a distributed index is created. This can also be compared to content discovery in peer-to-peer networks. Forestiero et al. proposed a descriptor sorting and replication algorithm for grids called *Antares* in [20]. The sorting is done by ant-inspired agents that travel the grid and pick and drop descriptors. According to a given probability an agent picks one or more descriptors. Descriptors most different from the rest of the descriptors on the current host are more likely picked.

Michlmayr [21] proposed a search algorithm for unstructured peer-to-peer networks. The so called *SemAnt* algorithm extends AntNet for query routing. A query is represented by a number of ants. At startup, SemAnt can be compared to k-random walk [22], however, backward ants spread pheromones and therefore the next forward ants can be guided to the resources. The limitation of this algorithm is that the content location is assumed to be static. Another example for search in peer-to-peer networks is called *AntSearch* [23].

Similar to the search problem, researchers adapted AntNet for routing purposes. Hossain et al. [24] propose two algorithms to route content in resource constrained networks. *Improved AntNet* is similar to the basic AntNet, except that forward ants track nodes they have already visited, i.e., perform cycle detection. *Pharaoh* additionally introduces negative pheromones if cycles are detected to avoid unnecessary movement of other forwarding ants. In our work cycle detection is implemented seamlessly.

Datta, Quarteroni and Aberer present an adaption to gossiping which makes information dissemination more selective and therefore reduce the number of messages in the network [25]. Their autonomous gossiping uses a stateless self-organizing algorithm. Similar to ant routing algorithms, the routing decisions are taken by the data items themselves. Guéret, Monmarché and Slimane modify the approach by sniffing the network and use information trails to build the necessary profiles for guiding the data items [26].

Jiang et al. describe an artificial immune network model based on the regulation of the endocrine system for solving combinatorial optimization problems such as the Traveling Salesman Problem (TSP)[27].

An example for multimedia and bio-inspired algorithms being merged is a content repurposing system introduced in [28]. The authors want to efficiently combine the right content with the right heterogeneous devices based on given network conditions. In traditional systems servers statically decide about repurposing chains and expect that the service nodes do not change their location or their service quality. The proposed *BioReSS* is a service selection alternative. It extends AntNet with quality of experience (QoE) and QoS metrics.

Balasubramaniam et al. [29] introduce a system for future content-based service environments, e.g., video on demand (VoD) services. The goal is a self-configuring system to easily inject services, represented by agents holding a specified energy level. The energy level is positively influenced by fulfilling user applications and negatively influenced by residing on nodes without action.

3. SElf-organizing Artificial HORmone SystEm - SEAHORSE

The heart of SEAHORSE is an artificial endocrine system (see [9]) where nodes are cells that create, distribute and act based on virtual hormones. These hormones are created (positive feedback) and travel around the body where they lead to specific actions in target cells (are consumed - i.e., negative feedback). The same hormone might lead to different actions, depending on the target cells [30].

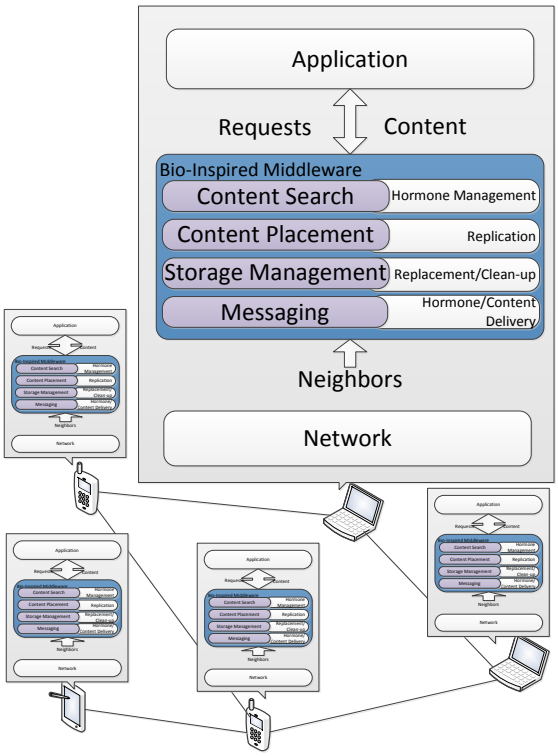


Figure 1: SEAHORSE Structure

In contrast to the ant-based systems described so far, we do not consider the typical forward and backward ants. Additionally, we assume that the location of resources is dynamic. Our idea is that the requester cell diffuses hormones leading to actions in other cells (nodes). Such an action is either to forward hormones or forward content if it is attracted by hormones in the neighborhood. Content organizes itself by placing replicas in the network to minimize the delay and increase the robustness. The replication mechanism uses local knowledge and in an ideal case replicas are at the right places before they are requested.

SEAHORSE can be seen as a middleware that hides the transport of content from the application and the application from the network. Figure 1 depicts an overview of a sample network with the installed system. Heterogeneous devices run the same middleware. The application interfaces allow for mobile and desktop-friendly implementations. The interface to the network allows for exchanging the messages on different network media. We assume that each node either knows or can figure out its direct neighbors. SEAHORSE can be applied on mobile and wired devices.

As it is hard to develop self-organizing algorithms properly on real devices instantly, usually, the behavior is first evaluated in a simulation. Note that the interfaces of SEAHORSE help the development process by separating the application from the network. The application can be independently developed and by using the proper interfaces, it can connect either to the simulator or to a real-world network. With the help of the simulator, system designers are able to analyze the network behavior with thousands of nodes and tweak the configuration appropriately.

In SEAHORSE we speak of *units* to be distributed by an application. A unit can be considered as a small piece of information (e.g., a short video, video scene, picture, information, a piece of data). The notion of units is similar to the notion of chunks used in BitTorrent-like systems, but units can have different sizes and can be semantically meaningful (e.g., a video scene, an image, a small file). The application that creates the units has to define what a unit actually is. SEAHORSE allows to compose arbitrary requests consisting of a number of units. The units can be composed sequentially (e.g., a movie is composed of a sequence of frames), in parallel (e.g., a number of images shown

in parallel), or mixed (e.g., a number of surveillance videos shown in parallel, each consisting of a sequence of frames). This supports personalized content consumption.

Application Interface

SEAHORSE does not spread requests such as typical systems, instead it uses hormones to indicate interest for each unit separately. This leads to emerging hormone patterns, allowing the nodes to decide more precisely about what units have to be replicated and which not (see Section 3). An application, however, has to interact with the user and there the notion of requests is common.

To make applications exchangeable, we specify a common language independent interface and select a notation where the client can define which content has to be delivered and also (if this is part of the application) of which quality. Furthermore, sequential, parallel and mixed compositions of units are implicitly supported, which could be also used by an application to present multimedia units in split-screens or traditional sequential formats. This description mechanism is called Video Notation (ViNo) [31].
Video is one of the most challenging content types as it has specific QoS requirements. However, any other type of content delivery can be described with it. E.g., a sequential large file is described as a sequence of content units that belongs together. The definition of a unit is application dependent and can thus be also a chunk of several KBs or MBs. The average size of a unit depends on storage available on nodes as well as the network setup, as larger units take longer to be replicated and transported (see [9] for a study of unit sizes).

ViNo is simple, but powerful and allows for applications in any programming language. To show the details, let us consider that a user wants to get two text-type files related to "skiing" and wants to see the results in a split-window. In ViNo such a request can be formulated as the following simple expression: $[u_1||u_2]$ where $||$ is the parallel operator that describes the split-window and u_1 and u_2 are identifiers such as file names. If a user does not know the exact file name, but only wants to use the tags "skiing" and "text" only, ViNo supports wildcards represented by $u_?$. The request would change to $[u_?||u_?]$. This expression does not contain the type and the tag, therefore ViNo allows to specify metadata and links it to the presentation by using a place holder unit u_0 with size 0. The resulting ViNo request is $u_0 \leftarrow_{tag=skiing,type=text} [u_?||u_?]$. An appropriate user interface of the application might help the user to formulate this request (see Section of use case 1 or [32]). The ViNo request does not leave the node, it is an independent communication interface between application and SEAHORSE. SEAHORSE translates the description into hormone concentrations and uses ViNo for delivering the content in the right order to the application.

Target networks

To support SEAHORSE, a network needs a minimum functional set consisting of:

- *Network awareness.* A network node must have access to a list of connected nodes to which it can send data. This list can either be set up before runtime or be created dynamically with some discovery protocol.
- *Link quality indicator.* A node needs to be able to assess the quality of each link. This is important for adjusting the hormone distribution according to the link strength. Link quality can refer to either physical qualities like signal-to-noise ratio or be described more abstract considering also the remaining bandwidth with respect to current load. The link quality is typically changing over time and needs to be measured periodically or on demand, respectively.
- *Local data exchange.* A node needs to be able to exchange data with its directly connected neighbors.

Typically, sensor networks have such characteristics, however, with some exceptions [33] these are usually not used for multimedia networks. Wireless mesh networks [34] are an interesting field of application, which typically support these functions as well and require an optimized usage of their bandwidth. Applications of such networks can be found for mobile computers in field operations as well as for some satellite networks like the Iridium communications system [35]. While the standard

internet transport layer protocols such as TCP or UDP do not offer network awareness, there exist several overlay networks offering location and network awareness [36][37] that are suitable target networks for SEAHORSE. Another possibility are overlay generation networks, like the scale-free overlay proposed in [38]. Wieser et al. [39],[40] propose Flocks, a mechanism that creates overlays according to given metrics, such as QoS. The algorithm allows for building a QoS map of the current network state, such that dynamics, as common in wireless networks, can be easily covered.

Artificial Hormone Delivery

Our work adapts the principles of the endocrine system of higher mammals. In the endocrine system glands such as the epiphysis create hormones. The hormones are released to the blood system and reach target cells, where specific actions are triggered. The actions depend on the type of the target cell. For example, the epiphysis creates melatonin that regulates rhythmic behavior such as the sleep-wake cycle. During the night melatonin is released and docks to specific brain cells. So, the positive feedback (darkness) amplifies the creation of melatonin, whereas the negative feedback (daylight) stops it [41]. Several artificial hormone implementations that follow these principles are described in [30]. In these works the artificial hormones are applied to distributed task allocation, robot swarms observing an area, etc. whereas the models differ in interpretation of the endocrine system. Therefore, we provide the definition of our proposed model as follows.

We define the artificial hormone system as a dynamic network of nodes, where the nodes are the cells creating and consuming hormones and the network represents the blood stream in which hormones and units can travel. Hormones indicate interest in a specific unit or in its content. So a requester periodically creates hormones and the rest of the nodes forwards hormones. The goal is to guide content through the network to the requester by moving content to neighbors with higher hormone concentration. Thus, a hormone path with the highest concentration at the requester has to be created. If a node gets hormones it keeps part of the hormones and forwards the rest to the neighbors. This can be considered as a natural TTL, because at some point there are no hormones left for forwarding. If a corresponding unit is found on a node, the node stops diffusing hormones and the unit starts its movement towards the requester. Further negative feedback is implemented by evaporation of hormones.

The algorithm is periodically executed by each node (see Algorithm 1) until a specified time stamp max is reached.

ALGORITHM 1: Execution loop of each node

repeat
  handle incoming requests
  diffuse hormones
  move content
  evaporate known hormones
  $timestamp \leftarrow timestamp + 1$
until $timestamp = max$;

We show how the algorithm handles requests, by the example of an incoming sequential request (see Algorithm 2). If the first unit is stored at the node it will be sent to the application, otherwise the corresponding hormone H_{u_i} is increased by η_0 or η. The values for η and $\eta 0$ are constant values configured at system start-up. Note that the position of the unit within the request also influences the amount of hormones to be created. Units at the beginning of the request get more hormones than the ones in the back and thus move more likely than more distant units (as shown by "position" in the last line of the algorithm).

ALGORITHM 2: Handle incoming sequential request

Input: generated request
if u_1 *is stored locally* **then**
  send u_1 to application
  $request \leftarrow request - u_1$
else
  forall the u_i *in request* **do**
    if $request = new$ **then**
      $H_{u_i} \leftarrow H_{u_i} + \eta_0$
    else
      $H_{u_i} \leftarrow H_{u_i} + \eta$
    $H_{u_i} \leftarrow H_{u_i} * (length(request) - position_{u_i})$
  end

The diffusion of hormones is performed as shown in Algorithm 3. A node decides whether to store a unit for the collected hormones. If a corresponding unit is not found the node forwards hormones to the neighbors (messages are sent only if the unit is not present). The forwarded hormones $H_{i_{Diff}}$ are a fixed percentage (α) of the existing hormones H_{u_i}. Each neighbor N_j gets one part of $H_{i_{Diff}}$ depending on the provided QoS weight w of N_j (e.g., influenced by RTT, link quality, etc.). Thus, a neighbor that provides better QoS gets more hormones than another. This influences the path units will travel, since they are attracted by higher hormone values. If a unit is found on the current node, the demand for this unit is obviously fulfilled and the hormones can be deleted. Note that at some point no hormones are left to be further diffused (i.e., the value of hormones represents a TTL).

ALGORITHM 3: Diffuse hormones to neighbors

forall the H_i *in stored hormones* **do**
  if u_i *for* H_i *is stored locally* **then**
    $H_i \leftarrow 0$
  else
    $H_{i_{Diff}} \leftarrow H_i * \alpha$
    $H_i \leftarrow H_i - H_{i_{Diff}}$
    forall the N_j *in neighbors* **do**
      $H_{i_{N_j}} \leftarrow H_{i_{Diff}} * w$
      forward $H_{i_{N_j}}$
    end
  end
end

The next step is to take care of the guidance of units which is shown in Algorithm 4. A unit will move to the neighbor with the highest hormone value max_{H_i}. To migrate a unit to this neighbor, max_{H_i} has to be larger than the local hormone plus a migration threshold m. The replication works as follows. If a unit is currently in use or it is popular in the neighborhood (indicated by the aggregated hormone concentration) it will be copied, otherwise moved (we have investigated different replication and storage balancing techniques in [7], [8], [9]). Before transport, units are collected in output queues. Each node has one output queue per neighbor. These output queues are sorted, i.e., a unit for which a higher hormone concentration exists is favored.

ALGORITHM 4: Move units

```
forall the u_i in storage do
    if u_i in use by the application OR popular in the neighborhood then
        copy ← true
    get maximum max_{H_i} hormone from neighbors
    if max_{H_i} > H_i + m then
        if copy then
            copy u_i to output queue towards this neighbor
        else
            move u_i to output queue towards this neighbor
        end
    end
end
```

The final step is the evaporation of hormones which is described in Algorithm 5. The evaporation comprises two tasks; (1) reducing all known hormones by a fixed value ϵ and (2) deleting all hormones with a value below the fixed threshold t. This guarantees that unused hormones disappear in the course of time.

ALGORITHM 5: Evaporate known hormones

```
forall the H_i in stored hormones do
    H_i ← H_i − ε
    if H_i ≤ t then
        delete H_i
end
```

Parameters

We presented the model by using a number of parameters that can be configured before system start-up. These parameters are listed in Table 1. The number of parameters allow for a wide applicability. The value range is between 0 and 1 for α and positive for the rest of the parameters. The advantage of having many possibilities for supporting different applications and networks also comes along with the disadvantage that system designers have to configure these parameters.

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 later on at the requesting node until the request is fulfilled. α is the percentage of hormones that can be diffused to neighbors. The evaporation value ϵ will be subtracted for reducing the hormones on alternative paths. These parameters need to be tuned. 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 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. The parameter b describes the time a transport has to take to consider a link as busy.

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. In our previous work we optimized them offline, using an evolutionary algorithm as described in [42]. We have chosen this offline configuration with pre-profiling to avoid monitoring costs at runtime. Especially, when targeting QoS-sensitive content such as multimedia, QoS violations have a high impact on the user experience.

The offline evolutionary algorithm initially creates a random population of parameters. Then, it 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

Table 1: Parameters to configure at system start-up

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
b	Minimum time needed a link is considered as busy

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.

We used the evolutionary algorithm 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. 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.

As this works for a single application of the artificial hormone system as shown in our previous works, a middleware (SEAHORSE) requires a more generic approach.

When designing the algorithm we wanted to have the highest possible flexibility regarding parameter values, which however made the system hard to configure. To reduce the configuration complexity, we (1) reduce the number of parameters and (2) discuss the settings of the remaining parameters in two case studies.

With a reduced number of parameters the evolutionary algorithm can be skipped and manual tweaking is again possible without major effort. For the following discussion, we rely on our experiences with the algorithm and the numerous experiments we applied. For validating these measures, we re-executed some of our experiments from [6], [7], [9] and [8] including different replication strategies and node churn and did not see any performance degradation. In the later sections we show by two different use cases that the simplified configuration is valid for applications of diverse fields.

The first measure of simplifying the parameter setup is to normalize the value range of the parameters (i.e., between 0 and 1). This simplifies the setting of parameters manually and does not change the relation of the parameters to each other.

η and η_0 are used to describe interest in specific content and to set up the path from requester to resource. As the simulation takes usually several circles, η dominates the hormone creation and the influence of η_0 is negligible. This is an indication to merge these two parameters and use η only as a further reference. η indicates not only the interest in content but has also an influence on the available search space. Usually, the whole network should be covered, which leads to high values for η. With the normalization of values η started to converge to 1 for medium to large network sizes (from 100 up to 10,000 nodes). This indicates that the value of η can be fixed to 1. The minimum hormone strength t helps to evaporate hormones faster and to get rid of non required alternative paths. This value would be clearly very low and in relation to ϵ and η negligible. In our experiments we have shown that t can be covered by ϵ. Hence, we set the value of t to 0.

m, the migration threshold, is used to drive unit movement. If m is high, a unit is likely to stick on the current node and requires more units to move to another node. This should avoid useless movement of units, but would never be very high, as the requester would have to wait too long for being served. In our experiments this value usually tends towards 0 and we therefore decided to set

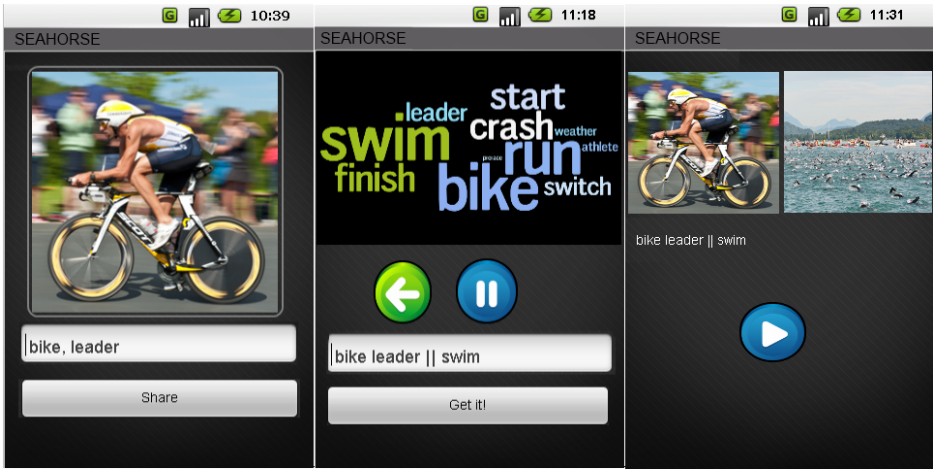


Figure 2: Example application for multimedia sharing at the Ironman triathlon (1) produce content (2) request content (3) consume content

it to 0.

The α and ϵ and b parameters control the path of the unit transport and react to application specific settings such as unit sizes and network setups. Therefore, they have to remain configurable. For example, if the application comprises very small units, units might distribute over longer distances in shorter time (of course depending also on the network capabilities). So it is likely to have a lower α than for applications considering large units. In case of high churn rates the evaporation rate might be lower to increase the robustness of the system by allowing several alternative paths.

This set of three parameters per application can be configured manually, as we show in the following sections.

4. Case Study 1: Dynamic Multimedia Sharing at Social Events

At social events masses of devices are used. Visitors are taking photos and videos with their mobile phones or tablet computers while moving around the area of the event. By using SEAHORSE visitors are able to share their most interesting multimedia content with other visitors during the event without the need of specific infrastructure. As shown in [43] visitors cooperating and providing content for other visitors improve the experience of the event. Within such a scenario we have to handle masses of producers and consumers at different locations of the event who build a dynamic network.

We assume that an application provides a graphical user interface that allows the user to produce and to consume content. How such an application might be implemented is shown in Figure 2 [1]. Since the visitors are likely to produce short videos and a high number of photos, the application allows for tagging the content easily and provides predefined tags. SEAHORSE registers the newly generated content locally and if requested provides it to other nodes. Users can request content by composing a personalized presentation by using predefined or self-designed tags. In the middle of Figure 2 the user wants to see two units in parallel, one is tagged with “bike” and one with “swim”. The presentation is translated into a ViNo request and sent to SEAHORSE. When the content arrives the user can watch it over a simple presentation interface (shown on the right). This case study aims at showing the applicability of SEAHORSE for content delivery, i.e., the system evolves to a global equilibrium by stabilizing the placement of content units within short time. Secondly, we show how SEAHORSE can be configured for similar applications.

[1] image source: Ironman Austria 2011, gholzer, http://www.flickr.com/photos/georgholzer/

Parameter Analysis

To evaluate the impact on the parameters we use an open source simulator [2] that models the artificial hormone system with different replication and storage balancing models, different network types and client models. The nodes periodically perform the actions of Algorithm 1. We evaluate a scenario where at a certain point in time (simulation time 0) a high number of units are introduced. Although the clients continuously request content the placement of content stabilizes at some point.

We assume that clients are interested in composing their own presentations that comprise a number of units. So, the simulation of the client generates sequential requests consisting of a number of randomly chosen tags (representing streams of units). A request is fulfilled if a corresponding unit for each tag is available on the node. If one request is fulfilled the next one is sent. We assume that tags are application dependent and their similarity is predefined. Internally, we model a tag as integer values (e.g., "swim":1, "bike":2, "run":3, "leader":5, etc.) that can be combined as arrays (e.g. "bike" "leader" is [2,5]) like in a n-dimensional coordinate system. We define that similar tags should have similar positions in the coordinate system and therefore are comparable by calculating the Euclidean distance (e.g., swim and leader [2,5], and bike and leader [3,5] will have a low distance). Again, we assume that the definition of tags and their similarities are predefined.

The request generation is based on the client's taste (i.e., a predefined tag according to a Zipf-like distribution - cf. [44] for instant popularity). The taste might change during the simulation with a probability of 10 % after watching units. After each unit presented to the client the next unit should be available immediately or within 40 ms allowing for a continuous presentation of a sequence of units by 25 fps. Therefore, we introduce a corresponding deadline. If a deadline is missed, the client loses interest and no more hormones are created for that unit and the deadlines of the following units are updated. In this scenario we strive to minimize the number of missed deadlines, but do not present anything to the user if the deadline is missed. The video bit rate is set to 1 Mbit/s.

In this use case, SEAHORSE works on an overlay that builds a random network (scale-free networks lead to similar results for the examined network size [7],[9] and [8], therefore we do not consider them in this paper). We generate a connected Erdős-Rényi random graph for 50 and 500 nodes with an average node degree of 4. Since it is most interesting for us to show how the system behaves if many new units are injected, a node generates a number of units at system start-up. When a unit is generated its content is described by a tag. We assume that some tags are more popular than others, i.e., they follow a Zipf-like distribution. It is configurable how much of the limited space of a node is filled with units. In this case study a node can generate units until one 1 MB storage space is used, which leads to 400 units of 125 KB size in a 50 nodes scenario. The maximum storage of a node is 900 MB, which can be used for replicas or units in transport. For this scenario, we set the bandwidth between two nodes to 1 Mbit/s. Thus, if the algorithm is executed every second, the transport of one unit from one node to another takes theoretically one second or one time step. We repeat each simulation three times and average the results. The overall time of the simulation is set to 200 s, which is enough such that the system stabilizes as it will be shown later. Note that these settings support the simplification of the model to calculate the optimum delivery as described later in the document. We evaluated different unit sizes and realistic transport patterns including churn in [7],[8],[9].

As described in Section 3, we have reduced the parameters to α (diffusion rate), ϵ (evaporation) and b (busy link threshold). Using different parameter sets we show how the performance of the system is influenced by different network sizes, churn, etc.

We build a parameter landscape by using the simulator, where we executed the simulation with all possible combinations of the three parameters. The quality of a parameter set is evaluated with the fitness function already described in the parameter section, i.e., the fitness is the rate of successful requests in the end of the simulation. A request is successful if all of its units are delivered within the deadline.

The value of a parameter can be between 0 and 1 and changes with a 0.1 step size. Thus, the simulation was executed 3,000 times. In Figure 3, a 4D space is shown, where the x,y, and z-axis

[2] licensed under GNU GPL, http://code.google.com/p/videonetwork/, 2009-2012

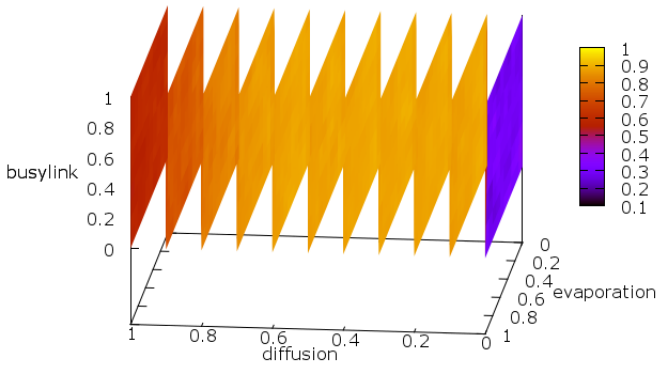


Figure 3: Parameter landscape with fitness displayed in the 4th dimension

represent the parameters and the 4th dimension shows the fitness as color. It can be seen that the parameter range for good fitness results is broad. The maximum fitness with 0.90 is reached with the configuration α=0.1, ϵ=0.1 and b=0.5. The average fitness of all runs is 0.77, where the median is 0.86, the 1st quartile is 0.74 and the 3rd quartile is 0.88. The average rate of missed deadlines is 0.001.

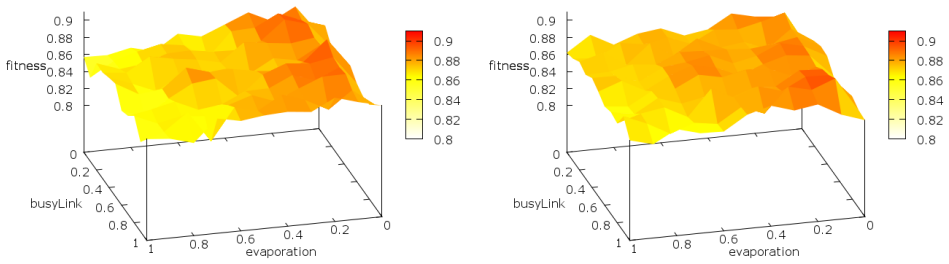
For α we recommend a value between 0.1 and 0.7. By setting α to 0 one can see that the the network can hardly do its work. If the diffusion parameter is too high, the needed hormone path is not steep enough for the requester to attract units within the given time. On the first sight the ϵ and b values do not have an impact on the fitness results.

Therefore, we simplify the visualization and reduce the number of dimensions by setting the diffusion parameter to a constant value. In the data set the parameters α with the values between 0.1 and 0.4 returned the best results. In Figure 4 it can be seen that this reduction helps to display the impact of the ϵ parameter. One can also see that the difference between the highest and the lowest fitness level is only 4 %. This means that the diffusion parameter is the most important parameter for setting up such a scenario. To set the value of ϵ we recommend a range between 0.1 and 0.3. A value of zero leads to the lowest fitness and thus proves that ϵ is necessary in the system.

To visualize the impact of b we set the ϵ value to 0.1, 0.2 and 0.3 and plot α on the x-axis and b on the y-axis. In Figure 5 one can see that if ϵ is set properly (e.g., to 0.1), many values for b lead to the highest fitness. If we set b=0 the resulting fitness is not considerably lower as if setting α or ϵ to zero. However, in Figure 5 (b) we see that b has an impact on the fitness, although the differences are in the range of 3 %. The highest fitness can be reached if b is set to 0.4 or 0.5.

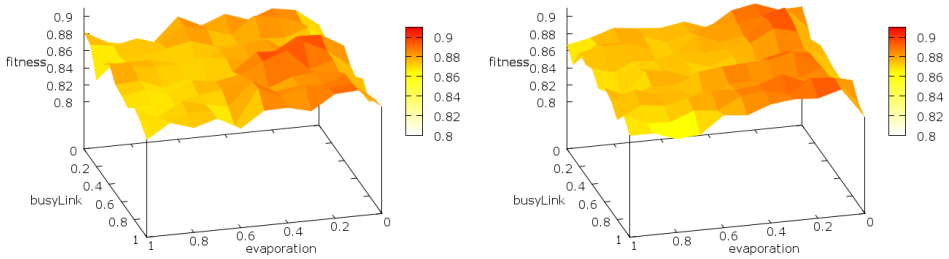
The fitness is a rather abstract measure, and therefore we show the impact of the parameters on the delay. If a request consists of a number of units, the delay for the first unit is measured from request time until arrival and can exceed 40 ms. For the other units, the delay is measured from planned time of playback until arrival. the delay is 0 if a unit is at the node before the playback should start. If a unit has missed its deadline then the delay for it is automatically set to 40 ms.

We compare the delay of parameter sets representing maximum, minimum, median and average fitness in Figure 6 (a). The delay development over simulation time is depicted as CDF plot for better visualization. We are specifically interested in the number of delay measurements below 40 ms and mark this value with the solid vertical line in the figure. We further increase the number of runs to 16 and the runtime to 500 s in order to get a confidence for the delay of 90 %. The plot shows that the difference between median and maximum is about 5 %, however, the performance drops if the fitness is below the average of 0.77. In all scenarios the placement of content stabilizes at some point in time, i.e., the delay approaches 0. The steeper the curve of the CDF, the earlier the delay



(a)

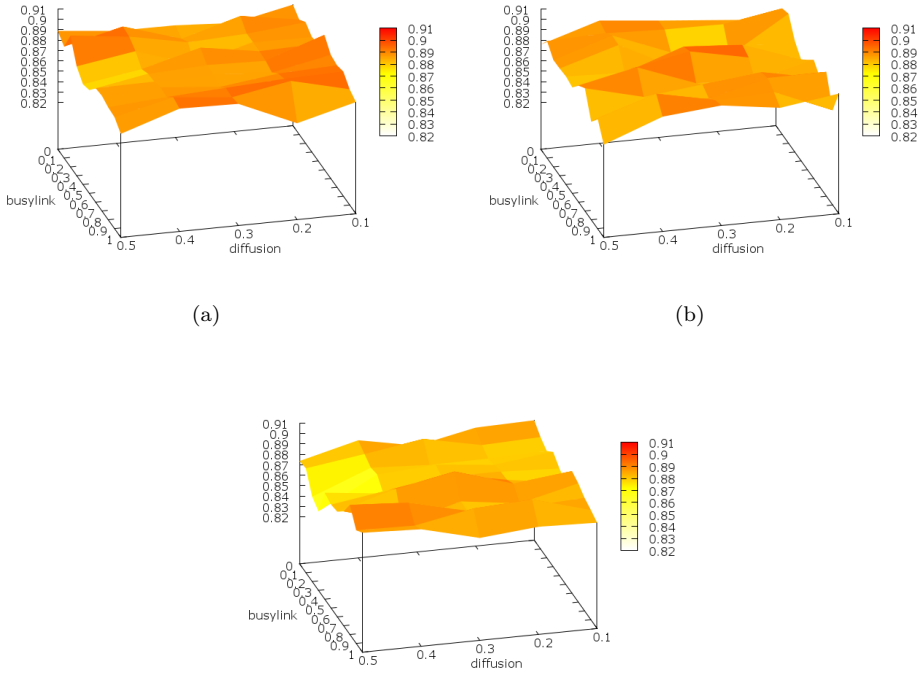
(b)



(c)

(d)

Figure 4: Parameter landscape if α is set to (a) 0.1, (b) 0.2, (c) 0.3 and (d) 0.4



(a) (b)

(c)

Figure 5: Parameter landscape if ϵ is set to (a) 0.1, (b) 0.2, (c) 0.3

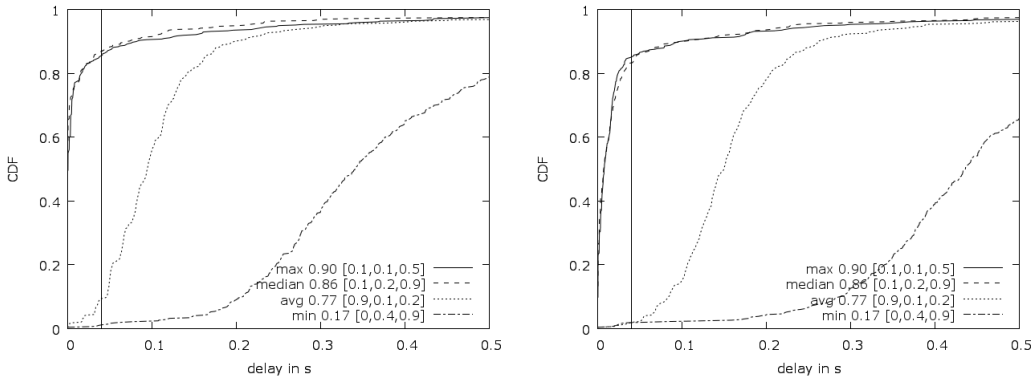


Figure 6: Delay distribution if parameters with min, average, median or max fitness is chosen, (a) no churn (b) 20% churn

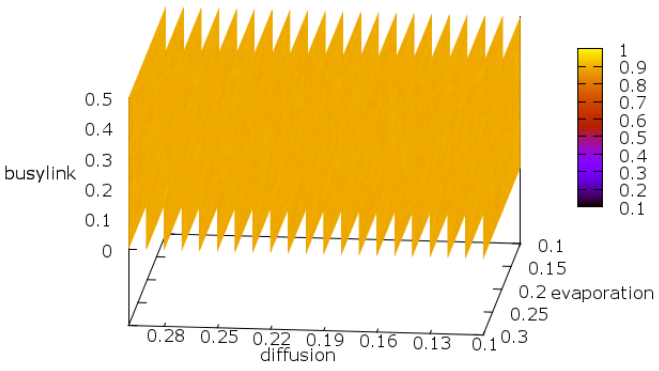


Figure 7: Parameter landscape with fitness displayed in the 4th dimension and a step size of 0.01

stabilizes. Note that these values include the delay of the first unit, i.e., the start-up delay. The average delay without the first unit is with the maximum fitness 2 ms, the median fitness 2 ms, the average fitness 4 ms, and the minimum fitness 11 ms. So, the major impact of the parameter set regards the start-up delay of a request. The delay between two units within a request shows that smooth playback is possible.

Since we consider dynamic networks we investigate the impact of churn if we keep the parameter set. We use a random churn model, i.e., a node is added or removed regularly (both with equal probability). In Figure 6 (b) one can see that the delay of maximum and medium parameters is influenced slightly, whereas the performance of the minimum parameter set drops. The inter-unit delay without start-up results in 3 ms for maximum fitness, 3 ms for median fitness, 5 ms for average fitness and 13 ms for minimum fitness. We see that the churn has an impact on the start-up delay, especially if the parameters with minimum fitness are used. A proper parameter set is therefore necessary, but it is sufficient to use the recommended parameter range.

We further evaluate the impact of the granularity of parameter values by decreasing the step size to 0.01. To reduce the number of simulations for the parameter landscape we predefine the ranges of the parameters. We evaluate the range from 0.10 to 0.30 for α, from 0.10 to 0.30 for ϵ, and from 0.0 to 0.50 for b. Thus, the simulation was executed 135,000 times (with 3 runs and 200s length).

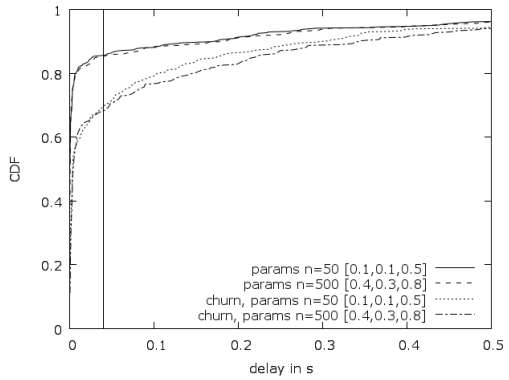


Figure 8: Delay distribution if parameters are chosen from the parameter landscape of 50 nodes or 500 nodes

In Figure 7, the results show a low variance of fitness values. The statistics of the value result in an average and median of 0.88, a first quartile of 0.88 and a third quartile of 0.89, the minimum value is 0.85 and the maximum is 0.91. Thus, it is not necessary to increase the granularity of the parameter values for obtaining better results.

Finally, we increase the number of nodes to 500 to show the scalability of the parameter set. In this case too, the parameter landscape leads to similar results as in the 50 nodes scenario, whereas the fitness values are in general reduced by 10 %. This can be explained by the increased number of open requests at the end of the simulation. The maximum fitness of 0.71 is reached if α is set to 0.4, ϵ is set to 0.3 and b is set to 0.8. The maximum parameters of the 50 nodes scenario leads to a fitness of 0.67. In the following we compare the delay of both parameter sets to show that the parameter set of the 50 nodes scenario is applicable if the number of nodes is increased. In this scenario we used 10 runs and a runtime of 500s, which is enough for a delay confidence of 92 %. According to Figure 8, the delay difference between both parameter sets is marginal. Additionally, the delay with a 20 % churn rate is shown. Again, both parameter sets have similar results, however, the delay rate below 40 ms is reduced by 15 %. The inter-unit delay for both scenarios is 3 ms with and 1 ms without churn. The delay is lower than in smaller networks, because the system has more units and it is more likely that the content of two units is described by the same tag.

Optimal content placement

We evaluate the proposed hormone-based method by comparing its performance to the best achievable placement for a representative network. In particular we compare average delay and request fulfilled rate at different time instants.

First, we examine the complexity of the problem. A formal model is developed with global view on the content placement. This model is applied in a centralized approach to find the minimum values of the performance measurements or to give at least lower bounds for them. The quality of the distributed algorithm implemented in SEAHORSE is validated by comparing its results to the optimal solution achieved by the centralized method. The centralized approach is applied for evaluation purposes only and it is not feasible to be implemented in real distributed video unit delivery applications. Therefore, there are strong accuracy requirements towards the optimization method but it does not have to comply with strict running time constraints.

The centralized model should comply with the same constraints as the distributed one but it should also have an exact knowledge about the current state of the entire system. Its input includes the network graph, initial location of the units, storage capacities of the nodes, link bandwidths, unit sizes and the series of requests. Similarly to the distributed case, no *a priory* knowledge is used on the future requests, therefore, each request is taken into account only after its appearance. No proactive replication is applied but passive, on-demand placement without any knowledge on the content popularity. All unit replicas are the result of previous deliveries: new replicas may appear at the destination and at any node along the delivery route.

NP-Completeness

The centralized model intends to deliver the video units from the nodes storing them to the requesting ones within the shortest time. This problem is similar to the edge-disjoint path problem (EDP) where a graph and a set of source-destination pairs (s_i, t_i) are given $(i = 1..k)$. The goal is to find edge-disjoint paths P_i for each pair such that path P_i connects vertex s_i to vertex t_i. In the content placement, the requesting nodes and the locations of the requested units represent the destinations and sources, respectively. The paths between them may overlap in our model but the paths are disjoint in time, i.e., the path segments occupied at the same time are edge-disjoint. We prove the $\mathcal{NP}$-completeness of the content placement problem by reducing EDP to it.

The complete proof is described in the Appendix. The $\mathcal{NP}$-completeness implies that there is no possibility of implementing a fast and exact algorithm for practical problem instances of particular size. This is why we concentrate on approximating methods in real-life problems. If the strict running time requirement can be neglected and really large problem instances are avoided, the optimal solution may still be found for validation purposes by using the tools of operations research and combinatorial optimization.

Solution method

We define an Integer Linear Programming (ILP) model of the centralized content placement problem to describe the optimization problem. The inequalities of the ILP model are generated from the input data describing the network, initial location of units and the request history. The detailed description of the model can be found in Appendix 7 (We discuss the accuracy of a more detailed and sophisticated version of this model in [45], here the ILP model does not consider replication).

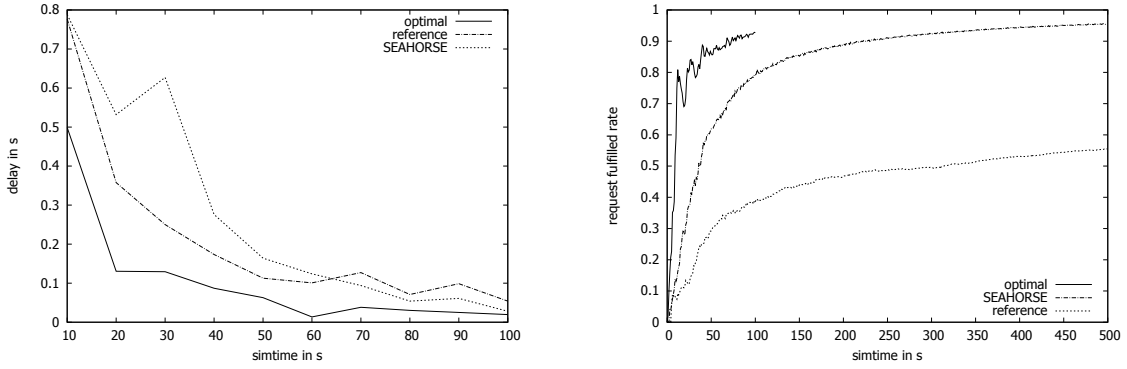
The model is quite complex and it uses at least 100,000 binary or integer variables for describing even a small size scenario, making this ILP task a real challenge. Initially, some preliminary analysis is executed including determining the distances between the nodes and the shortest time needed to fulfill requests possibly without missing any units. The results of this analysis are used for simplifying the inequality system. We apply an efficient solver called CPLEX[3] developed by IBM to solve the integer program. This calculation is a very time consuming step during the optimization due to the large computational complexity of the ILP problem. Post-processing is performed on the CPLEX result in order to gain statistical data regarding the optimal solution.

The requests are processed in an online manner and the algorithm takes the requests into account only after their appearance. The optimization is repeated for subsequent time intervals. Initially, the algorithm calculates the optimal unit delivery routes only for the set of units requested already at the beginning. Later, the algorithm proceeds step by step along the time line and it reevaluates the optimum by using an updated input set containing both the requests emerged at the current time and the units requested earlier but not having reached their destinations yet.

Better performance is expected from the centralized optimization than from a distributed one due to the global view, however, we are interested in how big differences are. Additionally, we compare a reference algorithm that uses iterative deepening and then delivers a unit hop-by-hop according the shortest path found, but does not replicate units.

For SEAHORSE we use the maximum parameter set determined in Section Parameter Analysis. The network size is 50 nodes, and at the beginning 1,849 units are created. We use the same settings both for SEAHORSE, the reference algorithm and the optimizer. In Figure 9 (a) the difference between the delay of SEAHORSE, the reference model and the optimal placement is depicted. In general we can see that the performance of SEAHORSE approaches the optimal delay after 70 s. Before that SEAHORSE leads to long delays, because hormones need to find a unit and then the unit has to follow the hormones to the requester. After 100 simulated seconds the delay stabilizes and tends towards zero because the placement of units improves. One has to note that a delay of 0 s also means that no messages have to be sent anymore (no hormones are diffused, nor content has to be distributed) because the units are already at the requester. The delay of the reference algorithm is better in the beginning, but never stabilizes such as the hormone algorithm (even not

[3]CPLEX, http://www.ilog.com/products/cplex/product/suite.cfm



(a) (b)

Figure 9: Delay and requests fulfilled comparison between optimal, reference and SEAHORSE delivery

after simulation times longer than 500 s). In Figure 9 (b) one can see that the fulfilled request rate (none of the units of a request missed its deadline) is much better for SEAHORSE than for the reference algorithm. SEAHORSE reaches a fulfillment rate of 80 % after 100 s in comparison to the 90 % of the optimal algorithm and the 40 % of the reference algorithm. The combination of hormones indicating interest with replication makes SEAHORSE successful in fulfilling requests. The delay of the reference algorithm is lower, because it does not need a stabilization phase such as SEAHORSE. The fulfilled request rate further indicates that many deadlines are missed by the reference algorithm. Thus, SEAHORSE is more successful in delivering within deadlines, but takes longer in delivering the first unit of a request where no deadline is set. In the start-up phase SEAHORSE can be improved by including active replication when new units are introduced in the system.

The centralized optimal model assumes an ideal case when the system is aware of the exact information on the state of delivery system including the network topology, the location of units and appearing requests. The different routes of units forming a requested sequence can be tracked and the delay between the units and the missed deadlines can be decreased significantly by synchronizing their deliveries. Furthermore, the centralized optimization may run several days while a system in real application will be required to take decisions within a fraction of a second.

5. Case Study 2: Information Dissemination in Smart Microgrids

For the second use case, consider a smart power grid, where households use smart meters to track the power consumption of electrical appliances. Intelligent smart meters such as introduced in [46] implement an event-mechanism that might be used to gather not only device information, but may be also used as a link to the providers. A mobile application that connects smart meters such as described in [47] can be used to collect price changes and distribute updates over the network. Then, either the user can be informed about the changes, and automatic device behavior can be triggered.

In this scenario we concentrate on the problem to reach all devices within the shortest possible delay. When using SEAHORSE, the application has to implement ViNo for expressing, e.g., the sending of a request for a specific update. The network has to provide a list of neighbors. Note that we use the very same delivery model as before.

Parameter Analysis

To evaluate the parameter influence on this application we use again the simulator mentioned in the first case study. We only implement a new consumer model where each consumer sends only one request for the same unit id. Additionally, we omit the deadline. Each time step new hormones

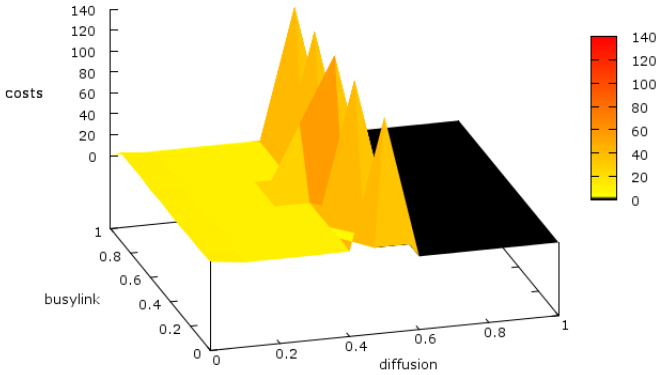


Figure 10: Parameter Landscape with ϵ set to 0

are created until the desired unit arrives. Note that this implementation does not touch the core of SEAHORSE, only a new client has been implemented because of different statistics to be collected.

We assume a random overlay network of 100 nodes as a basis and the rest of the settings remain the same as in the first case study.

In contrast to the first scenario we are only interested in those parameter sets that lead to full coverage, i.e., the information reaches all nodes. If not, we are talking about an invalid result. The fitness is calculated as the number of time steps (in s) required to reach the full coverage, the lower the number of time steps the better. Thus, actually we speak of costs for reaching full coverage instead of fitness. The costs can be calculated as $costs = p \cdot requiredtimesteps$, where p is 1 for full coverage and 0 for lower coverage after 200 s of simulation. The resulting costs are an average of 3 runs.

The resulting parameter landscape is shown in Figure 10. We have encountered that full coverage is only reached if ϵ is 0 and thus we reduced the visualization by one dimension. The results further show that α has to be between 0 and 0.6, otherwise the result is invalid. Note that if α is larger than 0.5, but smaller than 0.7, the coverage is reached, however, the costs are almost as large as the simulation time. The recommended values for α are therefore between 0.1 and 0.3. The busy link value b does not have a noteworthy impact on the costs, except a value of 1 leads to an invalid result.

According to the parameter landscape the lowest costs are reached if the parameter set is $\alpha = 0.1$, $\epsilon = 0$, $b = 0.1$. Since all nodes create hormones for the same tag, it is not necessary to configure a high diffusion value. Additionally, many alternative paths are needed, because all nodes want to get the unit. Hence, an $\epsilon > 0$ would be counter productive.

Our information dissemination scenario can be compared to pull-based epidemic spreading (gossip) such as introduced by [48] to ensure consistency in distributed databases. In pull-based protocols, a random node is chosen periodically and checked if it has the desired information. Pull-based protocols ensure full coverage, because if only one uninformed node is left, there is a 100 % chance that it probes an informed node. However, this can only be guaranteed if nodes from the whole system can be chosen uniformly at random. For the current evaluation the gossip algorithm assumes that all nodes are known from startup. In real implementations such as peer-to-peer systems, peer sampling protocols are used that manage the membership of peer-to-peer nodes dynamically, e.g., HyparView [49].

We integrated the gossip protocol into SEAHORSE as a new type of nodes. Its implementation required less than 100 lines of code, because the interfaces to the client and the network are well defined. Note that pull-based gossip has an advantage over the artificial hormone algorithm, which is the ability to directly connect to any of the nodes of the network.

Table 2: Time to full coverage comparison of pull-based gossip and SEAHORSE with different number of nodes

Nodes	SEAHORSE			Pull-based Gossip		
	min	max	avg	min	max	avg
100	10	13	11.7	12	14	13.2
1,000	16	19	16.7	16	23	18.9
10,000	21	23	22.1	23	26	24.7

In Table 2 we compare pull-based epidemic spreading with SEAHORSE as an average of 10 runs. We can see that the artificial algorithm used by SEAHORSE performs better than pull-based gossip, which shows the wide applicability of the artificial hormone algorithm [4]. If we increase the number of nodes (and keep the same parameter configuration for the artificial hormone algorithm), we can further see that the algorithm scales well, although the costs increase. With the help of ViNo it would be even possible to combine information patterns to orchestrate a number of appliances within a household.

The advantage of SEAHORSE is its modularity and if only three parameters have to be handled, the configuration effort is reasonable.

6. Conclusion

The design of self-organizing systems differs from traditional systems, because the overall behavior can just not be described by the sum of the individual interactions of all components. Typically, engineers take existing models and adapt them to their needs. Then, most of the implementations are dependent on a set of parameters that have to be well tested. The usual way is to implement a simulation. However, the step towards a real-world application requires a higher usability and well-defined interfaces.

In this paper we introduced SEAHORSE, a middleware that shows by example, how an existing self-organizing algorithm can be generalized. By specifying interfaces to the application the middleware transparently handles the distribution of content. We showed two use cases from different technical fields and performed a parameter analysis to reduce the configuration effort. The first use case is QoS driven and is complex as different content units are dependent on each other (the content is ordered). The second use case focuses on network coverage, hence the units are independent and much smaller. For both use cases the parameters were analyzed in detail, but we showed that a wide range of parameter values are applicable, e.g., different network sizes can be executed on the same parameter set (see 100 and 10,000 nodes in case study 2). Hence, manual configuration of the remaining parameters is possible and shows that SEAHORSE is general enough for self-organizing applications.

For the result evaluation of the first use case, we computed a centralized optimal distribution model and showed that the delay of SEAHORSE content distribution tends towards the optimum. Note that the centralized algorithm, which is used to get the optimum is not applicable in a real system because of the proven $\mathcal{NP}$-completeness of the problem model.

The second case study shows a very different application of SEAHORSE and we compare its performance with pull-based Gossip. SEAHORSE does not need full connectivity to support information dissemination and reach full coverage.

Simulations are necessary, because a distributed measurement system is hard to achieve, especially, if we want to evaluate multiple (thousands) of devices. We argue that SEAHORSE is a first step for bringing self-organizing algorithms towards real-world applications. Although it builds on a simulation, it can already interface with real applications with the help of ViNo.

[4] Note that we want to show the wide applicability of SEAHORSE to different domains and we do not focus on outperforming a more sophisticated combination of push and pull-based gossip algorithm.

Acknowledgements

This work was supported by Lakeside Labs GmbH, Klagenfurt, Austria, Alpen-Adria Universität Klagenfurt and funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214—17097—24774 and grant KWF 20214—18128—26673 and the Hungarian National Development Agency under grant HUMAN_MB08-1-2011-0010. We are further grateful for the support of Kornelia Lienbacher.

7. References

[1] L. Böszörmenyi, M. del Fabro, M. Kogler, M. Lux, O. Marques, A. Sobe, Innovative Directions in Self-organized Distributed Multimedia Systems, Multimedia Tools and Applications, Springer 51 (2) (2011) 525–553.

[2] M. Dorigo, M. Birattari, T. Stutzle, Ant Colony Optimization, IEEE Computational Intelligence Magazine 1 (4) (2006) 28–39.

[3] W. M. Shen, C. M. Chuong, P. Will, Digital hormone models for self-organization, in: Proceedings of the 8th International Conference on Artificial Life, 2002, pp. 116–120.

[4] A. M. Turing, The chemical basis of morphogenesis, Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237 (641) (1952) 37–72.

[5] W. Elmenreich, R. D'Souza, C. Bettstetter, H. de Meer, A survey of models and design methods for self-organizing networked systems, in: Self-Organizing Systems, Springer, Vol. 5918 of LNCS, 2009, pp. 37–49.

[6] A. Sobe, W. Elmenreich, L. Böszörmenyi, 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), 2010, pp. 3–8.

[7] A. Sobe, W. Elmenreich, L. Böszörmenyi, Replication for Bio-inspired Delivery in Unstructured Peer-to-Peer Networks, in: Ninth Workshop on Intellingent Solutions for Embedded Systems (WISES), IEEE, 2011, pp. 109–116.

[8] A. Sobe, W. Elmenreich, Replication and replacement in dynamic delivery networks, Complex Adaptive Systems Modeling, Springer 1 (1) (2013) 1–24.

[9] A. Sobe, Self-organizing Multimedia Delivery, Ph.D. thesis, Alpen-Adria Universität Klagenfurt, Austria (2012).

[10] F. Dressler, O. B. Akan, A Survey on Bio-Inspired Networking, Computer Networks, Elsevier 54 (6) (2010) 881–900.

[11] C. Prehofer, C. Bettstetter, Self-organization in communication networks: principles and design paradigms, IEEE Communications Magazine 43 (7) (2005) 78–85.

[12] A. V. Renteln, U. Brinkschulte, M. Weiss, Examinating Task Distribution by an Artificial Hormone System Based Middleware, in: 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 119–123.

[13] H. Hamann, J. Stradner, T. Schmickl, K. Crailsheim, Artificial Hormone Reaction Networks: Towards Higher Evolvability in Evolutionary Multi-Modular Robotics, in: Artificial Life XII (ALife XII), 2010, pp. 773–780.

[14] W. Trumler, T. Thiemann, T. Ungerer, An Artificial Hormone System for Self-organization of Networked Nodes, Vol. 216, 2006, pp. 85–94.

[15] A. Lindgren, A. Doria, O. Schelén, Probabilistic routing in intermittently connected networks, SIGMOBILE Mob. Comput. Commun. Rev. 7 (3) (2003) 19–20.

[16] G. D. Caro, AntNet: Distributed Stigmergetic Control for Communications Networks, Journal of Artificial Intelligence Research, AI Access Foundation 9 (1998) 317–365.

[17] M. Dorigo, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[18] S. Voulgaris, E. Riviére, A.-M. Kermarrec, M. V. Steen, Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks, in: In IPTPS06: the fifth International Workshop on Peer-to-Peer Systems, 2006.

[19] R. Kolisch, A. Dahlmann, The dynamic replica placement problem with service levels in content delivery networks: a model and a simulated annealing heuristic, OR Spectrum (2014) 1–26.

[20] A. Forestiero, C. Mastroianni, G. Spezzano, Building a Peer-to-peer Information System in Grids via Self-organizing Agents, in: 2nd Bio-Inspired Models of Network, Information and Computing Systems, IEEE, 2007, pp. 125–140.

[21] E. Michlmayr, Ant algorithms for self-organization in social networks, Ph.D. thesis, Technical University Vienna, Austria (2007).

[22] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and Replication in Unstructured Peer-to-Peer Networks, in: Proceedings of the 16th international conference on Supercomputing, ACM, 2002, pp. 84–95.

[23] K.-H. Yang, C.-J. Wu, J.-M. Ho, AntSearch: An Ant Search Algorithm in Unstructured Peer-to-Peer Networks, in: 11th IEEE Symposium on Computers and Communications (ISCC), 2007, pp. 429–434.

[24] M. Hossain, D. Elghobary, Ant based routing algorithms for resource constrained networks, in: Instrumentation and Measurement Technology Conference (I2MTC), IEEE, 2010, pp. 192 – 197.

[25] A. Datta, S. Quarteroni, K. Aberer, Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks, in: Semantics of a Networked World. Semantics for Grid Databases, Vol. 3226 of LNCS, Springer, 2004, pp. 126–143.

[26] C. Guéret, N. Monmarché, M. Slimane, Autonomous gossiping of information in a p2p network with artificial ants, in: Proceedings of the 5th international conference on Ant Colony Optimization and Swarm Intelligence (ANTS), 2006, pp. 388–395.

[27] H. Jiang, T. Liu, J. Chen, J. Tao, Endocrine-immune network and its application for optimization, in: Intelligent Computing for Sustainable Energy and Environment, Vol. 355 of Communications in Computer and Information Science, Springer, 2013, pp. 145–159.

[28] M. S. Hossain, A. El Saddik, A biologically inspired multimedia content repurposing system in heterogeneous environments, Multimedia Systems, Springer 14 (3) (2008) 135–143.

[29] S. Balasubramaniam, D. Botvich, R. Carroll, J. Mineraud, T. Nakano, T. Suda, W. Donnelly, Biologically inspired future service environment, Computer Networks, Elsevier 55 (15) (2011) 3423–3440.

[30] Q.-z. Xu, L. Wang, Recent advances in the artificial endocrine system, Journal of Zhejiang University SCIENCE C 12 (3) (2011) 171–183.

[31] A. Sobe, L. Böszörmenyi, M. Taschwer, Video Notation (ViNo): A Formalism for Describing and Evaluating Non-sequential Multimedia Access, IARIA International Journal on Advances in Software 3 (12) (2010) 19–30.

[32] M. del Fabro, K. Schöffmann, L. Böszörmenyi, Instant Video Browsing: a Tool for fast Non-sequential Hierarchical Video Browsing, in: USAB 2010, HCI in Work and Learning, Life and Leisure, Springer, Vol. 6389 of LNCS, 2010, pp. 443–446.

[33] D. Marcozzi, M. Conti, Image processing performance analysis for low power wireless image sensors, in: Workshop in Intelligent Solutions in Embedded Systems (WISES), 2007.

[34] B. Milic, M. Malek, Analyzing large scale real-world wireless multihop network, Communications Letters, IEEE (11) (2007) 580–582.

[35] C. Fossa, R. Raines, G. Gunsch, An overview of the IRIDIUM (R) low Earth orbit (LEO) satellite system, in: Proceedings of the IEEE 1998 National Aerospace and Electronics Conference, 1998.

[36] N. Manohar, A. Mehra, M. Willebeek-LeMair, M. Naghshineh, A framework for programmable overlay multimedia networks, IBM Journal of Research and Development 43 (4) (1999) 555–577.

[37] K. Zhang, LBA: Location and Bandwidth Awareness overlay for P2P live video streaming, in: International Conference on Networking and Digital Society (ICNDS), IEEE, 2010, pp. 327–330.

[38] I. Scholtes, Distributed Creation and Adaptation of Random Scale-Free Overlay Networks, in: 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 2010, pp. 51–63.

[39] S. Wieser, L. Böszörmenyi, P. Montessoro, Flocks: Evaluation of a qos- and content-aware overlay, in: Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on, 2013, pp. 33–40.

[40] S. Wieser, L. Böszörmenyi, Flocks: interest-based construction of overlay networks, in: Advances in Multimedia (MMEDIA), 2010 Second International Conferences on, IEEE, 2010, pp. 119–124.

[41] L. Rushton, The Endocrine System, Chelsea House, 2004.

[42] W. Elmenreich, G. Klingler, 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, IEEE, 2007, pp. 396–406.

[43] P. Peltonen, A. Salovaara, G. Jacucci, T. Ilmonen, C. Ardito, P. Saarikko, V. Batra, Extending large-scale event participation with user-created mobile media on a public display, in: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia (MUM), ACM, 2007, pp. 131–138.

[44] G. Dán, N. Carlsson, Power-law revisited: large scale measurement study of p2p content popularity., in: USENIX International Workshop on Peer-to-Peer Systems, IPTPS, 2010, p. 12.

[45] T. Szkaliczki, A. Sobe, L. Böszörmenyi, Discovering Bounds of Performance of Self-Organizing Content Delivery Systems, in: Eval4SASO Workshop of the 6th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO '12), 2012, p. 8pp.

[46] A. Reinhardt, D. Burkhardt, P. S. Mogre, M. Zaheer, R. Steinmetz, SmartMeter.KOM: A low-cost wireless sensor for distributed power metering, in: 36th Conference on Local Computer Networks, IEEE, 2011, pp. 1032–1039.

[47] M. Weiss, F. Mattern, T. Graml, T. Staake, E. Fleisch, Handy feedback: connecting smart meters with mobile phones, in: Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia (MUM), ACM, 2009, pp. 1–4.

[48] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, Epidemic algorithms for replicated database maintenance, in: ACM Symposium on Principles of distributed computing (PODC), 1987, pp. 1–12.

[49] J. Leitao, J. Pereira, L. Rodrigues, Hyparview: A membership protocol for reliable gossip-based broadcast, in: Dependable Systems and Networks (DSN), IFIP/IEEE, 2007, pp. 419–429.

[50] S. Even, A. Itai, On the complexity of time table and multi-commodity flow problems, SIAM Journal on Computing 5 (4) (1976) 691–703.

Appendix

In order to prove the $\mathcal{NP}$-completeness, the decision version of the content placement problem is used, where the problem is to decide whether there is a solution with total delay less than or equal to a specified value. The content placement problem is in $\mathcal{NP}$, since the suitability of a solution can be checked in polynomial time. The $\mathcal{NP}$-completeness of the content placement problem can be proved by reducing a special version of the edge-disjoint path problem (EDP) to it. Even et al. [50] proved that the disjoint path problem is $\mathcal{NP}$-complete even in the special case when the underlying graph is a directed acyclic graph (DAG). The nodes of a DAG can be grouped into hierarchical layers L_i in linear time by using e.g., the longest-path algorithm such that all edges from layer L_k point to layers L_l where $l > k$. In EDP, source destination pairs have to be connected with edge-disjoint paths while in content placement problems, the paths connecting the unit locations with the requesting nodes may overlap each other if they occupy a common edge during different time periods. In the following, we provide an algorithm on how to reduce an instance of the edge-disjoint path problem on DAGs to the content placement problem.

Construction:

Step 1 Take any instance of the edge-joint path problem on a DAG.

Step 2 Group the graph nodes into layers. Let L denote the number of layers of a graph. The layers are indexed from 0 to $L-1$. Edges may point from a layer only into another layer with definitely higher index.

Step 3 For each source node that is not located in the initial layer, add a node in the first layer and connect the new node with the source node by using a new edge. Replace the original source node with the new node. After that, each source is located in the first layer.

Step 4 If an edge connects non-adjacent layers, divide the edge into several consecutive edges with $k-1$ nodes where k denotes the difference of the indices of the connected layers. As a result, the edges connect neighboring layers only.

Step 5 Now let us introduce a content placement problem instance as follows. Let the directed edges be replaced with undirected ones. The modified graph represents the network topology graph. The modified sources (s'_i) and the unchanged destinations (t'_i) represent the initial location of the requested units and the requesting nodes, respectively. T denotes the sum of differences of the indices of the layers containing the requesting nodes and the corresponding destinations for each request. The storage capacity is infinite. All unit sizes and link bandwidths are selected in such a way that each unit can be copied on a link in one time unit.

Theorem 1. *The original EDP instance can be solved if and only if the constructed video delivery problem instance can be solved with total delay of at most T.*

Proof. First, let us assume that all source-node pairs can be connected with edge-disjoint paths in the original problem instance P. Each (s_i, t_i) path in P specifies a (s'_i, t'_i) path in the reduced instance P' as well. The (s'_i, t'_i) paths are edge-disjoint in time as well because the general disjointness implies disjointness in time. It's easy to check that the total delay is T. Thus, one direction of the proof is ready.

Now, it has to be proven that if the reduced problem instance P' can be solved with total delay of T then the original problem instance P has k disjoint (s_i, t_i) paths. The distance between two nodes is at least their layer difference. The value of T was selected in such a way that the total delay can be equal to T only if each (s'_i, t'_i) pair is connected along the shortest path running through layers with increasing indices. It has to be proven that the corresponding (s_i, t_i) paths are edge-independent. Each (s'_i, t'_i) path can reach a node in the time equal to its layer index. Therefore, disjointness in time implies that no overlap can be found on edges running between two neighboring layers. If overlap does not occur between any two layers then all the paths are also edge-disjoint. As a conclusion, the existence of a solution with total delay T in P' determines k edge-disjoint paths in P. □

Remark 1. *The proof implies that the content placement problem remains $\mathcal{NP}$-complete even if (1) each unit is requested from at most one node, (2) at most one unit is requested from a node, (3) there is no queuing in the solution and (4) there is no storage limit.*

ILP Model

In order to derive the integer programming formulation of the optimal content placement problem, we introduce the notations, variables and formulas as follows:

R denotes the set of requests $(0..|R|-1)$. N denotes the set of nodes $(0..|N|-1)$. E denotes the links between nodes $(\subset NXN)$. The topology of the network can be described by a directed graph $G(N,E)$. U denotes the set of units $(0..|U|-1)$. T denotes the set of time steps $(1..T_{MAX})$.

Constants:

- b: bandwidth of a link (same for all links) (in bit/sec)
- s_0: size unit, each size should be its integer multiple
- d_0: time step length, the unit of time resolution (in sec) ($d_0 \cdot t$ gives the time from starting the system to the tth time step). It is recommended to be the copying time of s_0 on a link $(= s_0/b)$
- p: playback rate (in bit/sec), the link bandwidth should be its integer multiple
- d_{max}: maximum delay between subsequent units in a composition (in sec) (it should be much less than the length of one time step: $d_0 > i \cdot d_{max}$ where i denotes the maximum number of units in a requested composition)
- w_R: weight of the units missed in the optimization goal. It should be larger than the maximum possible value of the total delays.

Unit related notations:

- $s(u)$: the size of unit u (in s_0)
- $t_c(u)$: the duration of copying unit u on one link (in time steps) ($= s(u)$, it is integer in case of appropriate parameter values.)
- $t_p(u)$: the duration of playing unit u (in time steps) ($= s(u) \cdot b/p$, it is integer in case of appropriate parameter values.)
- $B(u)$: set of nodes where unit u is initially stored

Node related notations:

- S_n: the storage size of node n (in s_0)
- $N(n)$: the set of adjacent nodes of node n

Request related notations:

- $I(r)$: the set of unit indexes in the composition queried by request r. $(0..|I(r)|-1)$.
- $u(r,i)$: the ith unit in the composition queried by request r
- $n(r)$: the peer where request r appears
- $t(r)$: the time of appearing of request r

Variables:

- $X_{u,n,t}$: (0, 1) variable to indicate whether unit u is located on peer n at time step t. $(u \in U, n \in N, t \in T)$

- $L_{u,n_1,n_2,t}$: (0, 1) variable to indicate whether unit u is copied/moved from peer n_1 to peer n_2 by time step t. $(u \in U, (n_1, n_2) \in E, t \in T \wedge t > t_c(u))$
- $Y_{r,i,t}$: (0, 1) variable to indicate whether unit $u(r, i)$ is served at time step t. $(r \in R, i \in I(r), t \in T \wedge t > t(r))$
- $R_{r,i}$: (0, 1) variable to indicate whether serving unit $u(r, i)$ is failed before deadline. $(r \in R, i \in I(r))$
- $T_{r,i}$: nonnegative integer variable to show the delay in serving unit $u(r, i)$. $(r \in R, i \in I(r))$

The problem of optimal content placement can now be formulated in the form as follows: minimize

$$\sum_{r \in R, i \in I(r)} T_{r,i} + w_R \cdot \sum_{r \in R, i \in I(r)} R_{r,i} \tag{1}$$

subject to

$$\sum_{u \in U} \left(s(u) \cdot X_{u,n,t} + \sum_{n_s \in N(n)} \sum_{t_i = t+1}^{min(t+t_c(u), T_{MAX})} s(u) \cdot L_{u,n_s,n,t_i} \right) \leq S_n, \qquad \forall n \in N, \forall t \in T \tag{2}$$

$$X_{u,n,1} = \begin{cases} 1 \text{ if } n \in B(u) \\ 0 \text{ otherwise} \end{cases} \qquad \forall u \in U, \forall n \in N \tag{3}$$

$$\sum_{n} X_{u,n,T_{MAX}} \geq 1, \qquad \forall u \in U \tag{4}$$

$$-X_{u,n,t} + \sum_{n_s \in N(n)} L_{u,n_s,n,t} + X_{u,n,t-1} \geq 0, \qquad \forall u \in U, \forall n \in N, \forall t \in T \wedge t > 1 \tag{5}$$

$$-(t_c(u) + 1) \cdot L_{u,n,n_t,t} + \sum_{t_i = t - t(c)}^{t} \cdot X_{u,n,t_i} \geq 0, \qquad \forall u \in U, \forall (n_s, n_t) \in E, \forall t \in T \wedge t > t_c(u) \tag{6}$$

$$\sum_{u \in U} L_{u,n_s,n_t,t} \leq 1, \qquad \forall (n_s, n_t) \in E, \forall t \in T \wedge t \geq t_c(u) \tag{7}$$

$$\sum_{n_t \in N(n)} L_{u,n,n_t,t} \leq 1, \qquad \forall u \in U, \forall n \in N, \forall t \in T \wedge t \geq t_c(u) \tag{8}$$

$$-(t_p(u(r,i)) + 1) \cdot Y_{r,i,t} + \sum_{t_i = t}^{t_p(u(r,i))} X_{u(r,i),n(r),t_i} \geq 0 \qquad \forall r \in R, \forall i \in I(r), \forall t \in T \wedge t \geq t(r) \tag{9}$$

$$-(t - t(r)) Y_{r,0,t} - \sum_{t_i = t(r)}^{t-1} X_{u(r,0),n(r),t_i} \geq -(t - t(r)) \qquad \forall r \in R, \forall t \in T \wedge t > t(r) \tag{10}$$

$$\sum_{t \in T} Y_{r,i,t} + R_{r,i} \geq 1 \qquad \forall r \in R, \forall i \in I(r) \tag{11}$$

$$-Y_{r,i,t} - \sum_{j=k+1}^{i-1} R_{r,j} + Y_{r,k,t-t_p(u(r,k))} + R_{r,k} \geq k - i - 1 \qquad \forall r \in R, \forall i, k \in I(r) \wedge i > k, \forall t \in T \wedge t \geq t(r) + t_p(u(r,k)) \tag{12}$$

$$T_{r,0} - \sum_t (t - t(r)) \cdot Y_{r,0,t} - d_{max} \cdot R_{r,0} \geq 0 \qquad \forall r \in R \tag{13}$$

$$T_{r,i} - d_{max} \cdot R_{r,i} \geq 0 \qquad \forall r \in R, \forall i \in I(r) \wedge i > 0, \forall t \geq t(r) \tag{14}$$

$X_{u,n,t} \in \{0,1\}$
$L_{u,n_s,n_t,t} \in \{0,1\}$
$Y_{r,i,t} \in \{0,1\}$
$R_{r,i} \in \{0,1\}$
$T_{r,i} \geq 0$

Let us take a look at the inequality system. The cost function consists of the number of missed units and the total delay. We combine the two optimization goals in a weighted sum where minimizing the missed units has priority over the delay. Optimization for total and average delay is equivalent since the number of unit requests is the same in all solutions to the problem instance.

Constraint 2 refers to the storage capacity of the nodes. It includes the storage need of the units whose copying already started but have not reached the node yet. Constraint 3 specifies the initial location of the units. Constraint 4 ensures that each unit is preserved until concluding the examination. According to Constraint 5, a unit can be stored on a node at a specified time step only if it was copied there at that time or was there in the preceding time step. According Constraint 6, a unit can be copied on a link if it was present at the starting node of the edge. At each time step, only one unit can be copied through an edge, see Constraint 7. In order to avoid multicasting, each unit can be forwarded from a node only into one direction (Constraint 8).

The conditions describe the unit distribution and copying in the network. Now, let us turn to the requests. Constraint 9 gives the most important condition of the request fulfillment at a specified time step, namely, that the requested unit should be present at the requesting node at that time and it should remain there while it is being played. Another important condition (10) for the first unit of a sequence is that the unit is not present on the requesting node before the time of its delivery. Of course, each unit of a sequence may be served at most one time step. It being never served it means that the deadline is missed (Constraint 11).

Condition 12 describes the relationship of successful deliveries of subsequent units within a composition: $u(i,r)$ denotes the ith member of a requested sequence r and let k denote the index of the last unit within the composition which was delivered before $u(i,r)$. In this case, either these two units are subsequent units in the composition or all requested units between them are missed the deadline. $u(i,r)$ can be delivered at a specified time t only if it is present at the end of the play of unit $u(k,r)$. In the formulation of this constraint we exploited that a unit is either present at the destination by the time of the end of playing the previous one or it misses the deadline. This is true in our case because of the arguments following. A unit may arrive only at the moment when a time step starts and never in the middle of it because new requests may appear only at the start of a time step and the length of a time step is equal to the transfer time on a link in our model. The allowed inter-unit delay (d_{max}) is so small that even if a series of units misses the deadline, the next unit is still late if it arrives in the next time step.

The remaining constraints refer to the delays. Constraint 13 specifies the delay for the first units of the sequences which is the difference between the arrival and the request time. Constraint 14 refers to the interunit delays which is either zero if the unit arrives in time or d_{max} if it misses the deadline. Other delay values cannot occur as it is proven in the previous paragraph.

If storage capacities are omitted, several simplifications can be introduced. Conditions on storage capacity (2) and unit preservation (4) can be omitted in this case. Unit replacement is unnecessary in the unlimited capacity case and it can be assumed that if a unit is present at a node it will stay

there. It can be exploited in simplification of some conditions 6, 9 and 10, see their modified versions below. Furthermore, two edges with opposite directions between the same two nodes don't have to be distinguished which leads to a significant reduction of the number of variables. All other conditions remain unchanged in the unlimited capacity case.

$$-L_{u,n_s,n_t,t} + X_{u,n_s,t-1} + X_{u,n_t,t-1} \geq 0, \qquad \forall u \in U, \forall (n_s, n_t) \in E, \forall t \in T \wedge t \geq t_c(u) \tag{15}$$

$$-Y_{r,i,t} + X_{u(r,i),n(r),t} \geq 0 \qquad \forall r \in R, \forall i \in I(r), \forall t \in T \wedge t \geq t(r) \tag{16}$$

$$-Y_{r,0,t} - X_{u(r,0),n(r),t-1} \geq -1 \qquad \forall r \in R, \forall t \in T \wedge t > t(r) \tag{17}$$