Organic Network Control Systems: Challenges in building a generic solution for network protocol optimisation

Bloch Matthias
Universität Passau
Institute of Intelligent Systems
Innstr. 43, 94032 Passau, Germany
bloch12@gw.uni-passau.de

***Abstract*—In the last years many approaches for dynamic protocol adaption in networks have been made and proposed. Most of them deal with a particular environment, but a much more desired approach would be to design a generic solution for this problem. Creating an independent system regarding the network type it operates in and therefore the protocol type that needs to be adapted is a big issue. In this paper we want to discuss certain problems that come with this task and why they have to be taken into account when it comes to designing such a generic system. At first we will see a generic architecture approach for such a system followed by a comparison of currently existing Organic Network Control Systems for adapting protocols in a Mobile Ad-hoc network and a Peer-to-Peer network. After identifying major problems we will summarize and evaluate the achieved results.**

***Index Terms*—Organic Network Control System, Organic Computing, Network Protocol Optimisation**

I. Introduction

Technical improvement is one of the biggest constants of the past decades. The devices we use every day are part of our lives and we can benefit from this improvement in many different situations where we do not even recognise it anymore. Not too long ago people had no access to the Internet, did not own a PC and actually had to go to a library if they wanted to have access to scientific literature. In our days mobile phones, wristwatches or even refrigerators can establish a connection to a local network, which gives them access to the omnnipresent internet. It is a privelege, which makes knowledge, data and services easily accessable for people all over the world. This interconnectedness has been built up over the years and is getting more complex with every year that passes by. Therefore improving this system is a task that has to deal with various components and we are reaching a point where, in some cases, a manual approach does no longer bring the desired reward.

This process can especially be observed when it comes to optimising processes that are already performing on a high level. In these scenarios people take a step back and rely on a technique that comes from nature itself. For example the human body is a product of a permanent self-learning, self-organising and self-optimising system, that evolved over thousands of years. From the technical point of view we try to apply these evolutionary processes to our environment to make it able to manage itself and improve at the same time.

This self-management is one aspect of a field called *Organic Computing* (OC)[12] and *Organic Network Control* (ONC) is an approach to use this technique for network protocols. These are used to establish a proper communication between different devices. They contain a certain amount of parameters that give information about different properties depending on the type of the network.

In this paper we collected data from current research and apply them to a generic approach of network protocol optimisation. We discuss certain problems that come with building a *Generic Organic Network Control System* and show up possible solutions. In Sec. 2 we present an architecture approach, show important components and give insight on the desired functionality. For better understanding of optimisation processes for different protocols we included two experiments in Sec. 3. In this section we go over describing the experimental setup, the protocols used and the results. After that we go into two main problems in Sec 4. In Sec. 5 we conclude this paper with a summary and an outlook to future tasks and challenges.

II. State of the Art

By far OC is getting used in multiple subject areas where wireless connections and networks in general can be found. Different approaches have been made adapting traffic control [1], *Clustering Algorithms* in mobile ad-hoc networks [10] and since 2005 the *German Research Foundation* runs a priority research program on OC [1]. When it comes to network protocols there are always particular solutions for certain network types. The problems emerge when it comes to optimizing these protocols, because the performance highly depends on the used parameter set. These differ between the different kinds of protocols and must be adapted individually for the current environmental needs so far. That´s the cause for a generic solution being very desirable to be able to solve this task a reasonable amount of time. Many different approaches have been made to solve this issue, but a real world solution does not exist yet.

A. Generic Architecture Approach for Organic Network Control Systems (ONC)

A fundamental thought for building and designing a generic solution for ONC is the architecture or the contained components of such a system and the requirements that have to be fulfilled to be able to dynamically adapt network protocols to various, unforseen situations. The architecture we want to set as a basis for further discussion of this topic was presented in [2,3] by Tomforde et al. and shown in Fig. 1.

The biggest problem with an architecture meeting this task is the requirement for the system to respond in a short amount of time to stay useful. Therefore the adaption needs to happen quick and this is the first challenge to be taken. The approach to solve this task for the presented architecture by Tomforde et al. in [2,3] is building up a layered system, with the first layer being a parametrisable Network Controller [2]. The layers responsible for the adaption are inspired by a generic *Observer/Controller* architecture presented by Müller-Schloer et al in [11], where each has different responsibilites in finding suitable solutions to environmental changes or requirements.

1) System under Observation and Control (SuOC): The SuOC is basically the node taking part in the network and is therefore using certain protocols to be able to communicate with other components connected to the network. In this architecture the SuOC takes over the role of the first layer (Layer 0) and needs to be a parametrisable network controller [2].

Layer 0 is responsible for processing the particular instance of a network protocol and collect basic information about i.e. used parameters. Further on for some networks, like *Mobile Ad-hoc*, the neighborhood of nodes is important so additionally for these the current status of the environment needs to be locally accessible and observable [2,8]. As this layer should always be exchangable it is never restricted to a particular set of protocols and has to provide, depending on the current protocol used, a set of variable parameters and a quality criterion for the performance measurement[2]. This criterion is also called fitness or evaluation function [8].

2) On-line adaption: The first adaptation step is situated at Layer 1 and adapts the network controller settings[2]. This component is able to react quickly with a solution from a predefined pool of classifiers (rule set) using the observed condition of Layer 0 as foundation for choosing an appropriate action.

At Layer 1 the observer additionally collects status informations and local settings of the current network protocol instance for further processing. The data is then made up with a performance prediction value and historical performance knowledge to build a classifier for a certain environment. By transforming it into an n-dimensional vector to provide a more normalized and abstract view of the situation, it is ready to be passed to the controller of Layer 1.[3]

The controller then decides, based on the vector received, if an adaption is needed[3] by a *Learning Classifier System* (LCS), which maps the input from the observer to a rule base of possible actions[2]. In particular the LCS builds a match set,

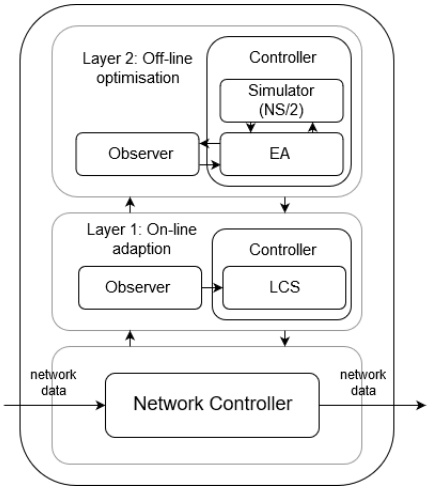

Fig. 1: Generic architecture approach of ONC [2,3]

which are all classifiers from the rule set matching the current environmental conditions. Then the contained classifiers are ordered and the most appropriate parameter set is applied to Layer 0. Additionally the performance with the new parameter set is predicted, to be able to calculate a reward by comparing the predicted value with the actual performance measure after the adaption. With this reward value the system is able to learn automatically by updating the contained classifiers appropriately to how they perform in certain situations.[3]

The System is running into an issue, if there is no suitable solution for a given situation. In this case the action with the most similar environmental condition is choosen by the *Covering Mechanism*, because the LCS is not allowed to create new classifiers by virtue of time restrictions[3]. If the rule set is empty the default action is chosen. The observed information is then passed to the observer at Layer 2, where the production cycle for new classifiers is started.

3) Off-line optimisation: The last layers responsibility is the optimisation of parameter sets. The *Off-line optimisation* has to solve the task to find a parameter set that is as close to the optimum as possible for a given protocol[2]. The two main components solving this issue are an *Evolutionary Algorithm* and a simulation tool, to be able to test a big amount of new created parameter sets in a closed environment. [3] Thereby the system is not affected and can operate simultaneously.

Additionally outsourcing this task to a new layer has a big advantage, because time restrictions are no longer as important and solutions can run various tests in the simulation before they are applied to the real world for the first time. Thereby the rate of bad performance and malfunction can be reduced [3] and the fitness function can be applied for new solutions, too. This leads to a more consistent performance of the protocol and a premature assessment of how the new parameter set will perform.

On this layer the Observer receives the system information from the observer at Layer 1 and passes it to the controller, where the *Evolutionary Algorithm* (EA) and the simulation

tool are located. After the passed vector is applied to the simulator for the first time the EA evolves multiple parameter sets and tests them using the simulator *NS/2*[2, 3]. The EA thereby is restricted by a pre-defined number of children per generation, because a higher number would lead to a higher quality of sets, but also to more time spent evolving them. The parameter sets that qualify and pass these tests are then transferred back as new solutions to Layer 1 and are applied to the real world on Layer 0.

B. Three major tasks of ONC

To be able to perform a precise performance measure and appropriately adapt the network controller to the current situation three major tasks have to be fulfilled: A textitperformance metric has to be defined [3,8] to be able to decide, by certain parameters, if an adaption is needed. Only this way we are able to have reliable indicators whether an adaption or a new protocol is perfoming better or worse.

Secondly a situation description accompanied by a distance function is required[3,8]. This is especially important for the LCS and the included *Covering Mechanism*. With this function we are able to decide if conditions are similar or not. For a generic solution a generalization of this distance function is required to be usable for multiple different scenarios. [8]

Thirdly a simulation model for Layer 2 needs to be developed [3,8], which includes e.g. creating different scenarios for testing.

III. Optimisation of network protocols: Peer-to-Peer vs. Mobile Ad-Hoc

A. Optimisation: Peer-to-Peer protocol

At first we want to look at the performance of the introduced system architecture when it comes to optimizing a *Peer-to-Peer* protocol. Both parts, the on-line network controller adaption and the off-line parameter set optimisation, were tested separately in this experiment. We will start with the experimental setup including the *NS/2* configuration, the used protocol and the used parameter set. The last part of this section will be the evaluation of both components' performance. This experiment was presented by Tomforde et. al. in [2].

1) Experimental Setup: For this experiment a standard PC (CPU: AMD Athlon 3200+, 2.00 GHz and RAM: 1.00GByte) was used taking part in a *Peer-to-Peer* (P2P) network. The "real-life"-scenario was inspired by a role model of a student in computer science during a day, looking up information sporadically on the internet, downloading data for university and additionally running a P2P-Client to download a high amount of data. The target set for this experiment was to download the data as fast as possible without interferring other processes or decreasing usability. In this case CPU and RAM remained unconsidered, because the model of the course of the day only contained upload-bandwidth and download-bandwidth.

Tomforde et al. [2] were measuring the off-line optimisation by comparing it with the usage of the standard parameter set and the on-line adaption was evaluated under the assumption that varying user behaviour can not be forseen completely, which was simulated by the previously described scenario.

As presented in the architecture approach before a *NS/2*-simulator was used as it is a standard solution for network simulations. The scenario was used as basis for the *NS/2* and was defined with the following configuration: number of nodes: 100, number of seeds(peers initially providing the data): 3, size of files: 500MB, and number of files: 1. The maximum possible link download-bandwidth was set to 400 KByte/sec and 40 KByte/sec for upstream.[2]

Before we go over to the evaluation we have to take a closer look at the used P2P-Protocol and its parameters. The used P2P-Client was based on the BitTorrent protocol, but Tomforde et al. [2] were using an implementation of a *BitTorrent*-like protocol, with some simplifications as presented by Eger et al. in [5]. The standard *BitTorrent* protocol was developed to increase the efficiency and reduce free-riding by dividing data into small parts called chunks. Each peer is able to download chunks from other peers, but when the download is finished the chunk is instantly uploaded to other peers. In difference to the standard *BitTorrent* protocol the used implementation does not implement a specific version, but it still behaves like the standard protocol. The *BitTorrent*-Client offers multiple parameters which might affect the performance, but Tomforde et al. [2] restricted the optimisation to seven parameters, which can be found in Table I with the used standard values.

Variable	Description	Standard value
NumberOfUnchokes	Number of unchoked connections	4 conn.
ChokingInterval	Interval for unchoking process	10 sec.
RequestPipe	Number of simultaneous requests	5 req.
NumberPeersPerTracker	Number of requested peers	50 peers
MinPeers	Min. Number of peers for not re-requesting	20 peers
MaxInitiate	Max. Number of peers for initialisation	40 peers
MaxConnections	Max. Number of open connections	1000 conn.

Tab. I: Variable parameters of the *BitTorrent* protocol [2]

2) Evaluation of Off-line optimisation: The basis for this optimisation builds the *BitTorrent*-based *NS/2* simulation and an *Evolutionary Algorithm* using the values listed in Table 1. Tomforde et al. [2] collected average values over multiple runs in which the fitness function always was defined as minimising the download time for the particular node.

A big influence was identified, when the number of peers was changed (5, 10, 50, 100), because for a higher number of peers a much bigger performance increase was recognized when optimizing the protocol. For a small amount of peers only a 1.1% increase in performance was observed. In contrast for an amount of 100 peers the performance increased by 14.67%. Looking at the optimized protocol parameter set for 100 peers (Tab. II), it gets clear that some parameters seem to have a major influence on the performance while others do not have changed at all. Tomforde et al. [2] explain this

behavior by the strict definition of the scenario, because it mainly uses constant connections, which e.g. lead to a rare usage of the un-choking process. Another example is the increase of the *MaxConnections* parameter. This adaption is based on the constant connections used, too, as in a more stable network more connections can be left open, because topology will most likely not change in the future. This leads to a better interconnectedness of the network and improves the performance of the network, because the steps of establishing connection to another node can be skipped if the connection was left open.

Variable	Standard value
NumberOfUnchokes	3 conn.
ChokingInterval	20 sec.
RequestPipe	8 req.
NumberPeersPerTracker	24 peers
MinPeers	20 peers
MaxInitiate	40 peers
MaxConnections	1104 conn.

Tab. II: Optimised solution of the BitTorrent protocol [2]

Summarizing the results, Tomforde et al. [2] are pointing out, that optimised parameter sets can lead to a performance increase of 0.1% up to 30.0% in specific configurations. The time needed to develop these optimised solutions is highly dependent on the number of peers and the file-sizes of the data. For a small number of peers and file-size (e.g. 5 peers, 5MB files) the evolution only requires a couple of minutes, but for a big number of peers and file-size (e.g. 100 peers and 1000MB files) it lasts about 24 hours until one simulation run is finished.[2]

3) Evaluation of On-line learning system: As we have seen in the section before the textitOff-line optimisation can lead to a big performance improvement by creating optimised parameter sets. In contrast to the Layer 2 component the On-line learning system needs to instantly react to environmental changes and appropriately use given parameter sets for specific situations. Every three minutes the ONC checks for changes and applies a new parameter set to the network controllers protocol if needed. We are starting with an empty rule set and therefore only the default parameter set can be used until the *Off-line optimisation* has finished developing new parameter sets. [2]

This is one of the main causes, why we can not see a big improvement on the first day, because the *Off-line optimisation* is still busy developing new parameter sets for Layer 1. These requests get queued up by the On-line learning system as soon as a new situation appears and therefore the average download-rate does not exceed the performance of the standard protocol (165.5 KByte/sec). On Day 2 a performance increase is achieved, which can be explained by a situation-depending selection of parameter sets. The average download-bandwitdh increased by 7.2% compared to the standard configuration (177.4 KByte/sec), because appropriate parameter sets for previously occurring situations have been developed and used in this particular situations. Tomforde et al. [2] point out, that the increase also can be influenced by the *Covering Mechanism*, simply on basis of more parameter sets being available to choose from. The last day shows the biggest improvement, because the *Off-line optimisation* added more possible parameter sets again. Now the *On-line adaption* is able to always choose the optimum parameter set and therefore the performance improvement lies by 20.4% (199.3 KByte/sec) compared to the standard configuration. Tomforde et al.[2] were not able to produce a better performing solution after day three, which leads to the assumption that for this scenario this is the optimum in performance. On Layer 1 the textitCovering Mechanism is also responsible for the performance improvement, because in situations where no particular solution was developed so far the most appropriate one is choosen this way.

The results of Tomforde et al. [2] are impressive and show that an ONC can successfully adapt network protocols to perform more efficient. On our way to a generic solution we will look at another example for adapting network protocols to be able to identify certain similarities or problems.

B. Optimisation: Mobile Ad-hoc network protocol

Mobile Ad-hoc networks are dynamic networks with at least two nodes that are connected through a wireless connection. It especially stands out by its mobile nodes and its constantly changing topology due to that mobility. Therefore special protocols are needed for such networks and its behaviour differs based on the situation. For example in a network with quick moving nodes probing for neighbours and refreshing the current topology is much more important than in a network, which consists of pretty stationary nodes.[6]

The following experiment was presented by Montana et al. in [6] and describes a comparison of automated and manual optimisation of *Mobile Ad-hoc* network protocols and the performance of an adapted dataset in an unknown environment. In this case it is a ”proactive link state protocol”, which is a specialized version and gets optimised in a scenario with moving nodes. The previously described architecture is not used, but we can describe important aspects and requirements for a generic solution through this example of network protocol adaption by a *Genetic Algorithm*. It is comparable with only having the Layer 2 of the previous example, but now operating in a non-static network.

1) Experimental Setup: The following experiment was based on a live mobile network demonstration performed for the *DARPA/Army Future Combat System Communication Program* at Lakehurst, NJ. On one side real world data from Lakehurst was used to compare manual with automated parameter tuning and additionally artificial datasets were generated by a scenario generator for a more controlled investigation on the optimisation process.

The simulation contained multiple mobile nodes (SUVs) and a helicopter being connected in a *Mobile Ad-hoc* network moving through a simulated, wooden terrain. Basis for this simulation was the *OPNET Modeler* with the *OPNET Wireless Module*, which allows the user to specify certain properties for the desired network simulation.[7] The scenario generator

was used to simulate 20 nodes moving through a square area of about 1200 meters length. The nodes were moving in a straight line to randomly selected waypoints with constant speed. From this general settings Montana et al. [6] created two datasets with a length of 100 seconds each. The difference between these two was the movement speed of the nodes. In one dataset the nodes were travelling with a speed of 10 meters per second and in the other one they were moving with 0.5 meters per second. One of them was used for evaluation during the optimisation process of parameter sets, and the other one was there to test the performance of already adapted parameter sets.

As they were operating in a *Mobile Ad-hoc* network, downloading-bandwidth was no suitable measure for the performance of a protocol, because of the fact how a *Mobile Ad-hoc* network works. When a node is taking part it can show its own presence by a periodic broadcast of heartbeat packets, which are received by another node in the network. If this second node detects multiple heartbeats in a short time a link exists between these two nodes. As long as these heartbeat packets are received the node is an active node in the *Mobile Ad-hoc* network. When no packages are received from a node, the link is considered as broken and the node is no longer taking part. These packages are distributed through all out the network with the effect of every node knowing about the whole topology. This distribution through the whole network is using capacities and makes detecting the download-bandwidth or upload-bandwidth unnecessary. Montana et al. have selected the dropped package score as the main indicator for performance measure.

To opitmize this score the *Genetic Algorithm* was allowed to modify following parameters of the used protocol: *Heartbeat Interval, Heartbeat Points, Score History Size, Up Score Threshold, Down Score Threshold, Routing Algorithm, Routing Event Interrupt Period, Routing Global Interrupt Period, Traffic Max Attempts*.

Variable	Explanation
Heartbeat Interval	Time interval for sending heartbeats
Heartbeat Points	Number of nodes to forward a received heartbeat to
Score History Size	Time window to observe for making decisions about scores/points
Up Score Threshold	Minimum value of score needed to establish a link to a neighbor
Down Score Threshold	Score must always be bigger, otherwise the connection is torn down
Routing Algorithm	Hazy Sighted Link State or Standard Link State
Routing Event Interrupt Period	Interval formodule checking for link changes
Routing Global Interrupt Period	Interval for node to sending a link state update
Traffic Max Attempts	Number of retransmission attempts

Tab. III: Parameters of used protocol for optimisation

To not have multiple parameter sets every node was using this set of parameters. [6]

Now we want to take a closer look at how Montana et al. in [6] have defined or restricted the *Genetic Algorithm* for optimising the parameter set. The representation of parameters was defined as real-valued. The big difference to a binary valued representation is that a maximum value, a minimum value and steps between the values have to be defined for all parameters. For generating new individuas a standard mutation and uniform crossover operators with a probability of 0.5 were used. For steady replacement an exponential parent selection was choosen, which means if a new individual was generated it was instantly added to the population and the worst individual in the population was erased. When an individual was adapted the simulation was triggered with the current configuration of the individual. During the simulation values about the network performance were gathered and combined to a single score.

The experiment started at a population size of 300 random individuals with a weighted scoring function defining the fraction of dropped packets and the average delay in seconds. As we mentioned before the dropped packets were the main concern, so it was weighted 10 times more than the average delay. Montana et al. [6] reported a big issue concerning the simulation time for a new parameter set. The simulation took about 7-8 minutes per individual of the Lakehurst dataset and slightly less for a generated scenario. Hence, the time for a full optimisation run exceeded multiple days on a single machine, because about 600-1000 [6] individuals had to pass the simulation. Therefore Montana et al. restricted the textitGenetic Algorithm to do a single run per experiment and were forced to waive getting average values for their optimisation process.

2) Evalutation of results: For the Lakehurst dataset Montana et al. [6] evaluate how effective the training is regarding the performance on the training dataset. They compared dropped packets and average delay for manually and automatically adapted datasets. For the artificially created data of the simulation generator the major concern is how good the adapted parameters perform in one or multiple known or unknown scenarios. To achieve this the training and later performance test are executed on different datasets.

The result of the comparison between manually and automatically tuned parameters of the set clearly shows that automated adaption outperforms manual adaption. In Table IV we can see what Montana et al. [6] discovered in their experiment in values.

Approach	Dropped Packets	Average Delay
Manual	0.089	0.0089
Automated	0.042	0.0036

Tab. IV: Results of Lakehurst dataset optimisation [6]

In their article they point out, that even without a directed optimisation and only on a basis of 300 random individuals the *Genetic Algorithm* performs much better than the random search algorithm, because it still discovers new solutions when

the random search is producing new sets either slowly or is no longer able to create new ones. [6]

Next they test the performance of optimised parameter sets in a simulation with currently unknown scenarios. For example an individual was trained on a dataset with high speed of the single nodes and now has to perform on a dataset with low speed of single nodes. Thereby they pick up a very important requirement for algorithms to perform well in the real world. "Generalization to new scenarios is critical to the success of networking algorithm in the real world."[6] The big issue is that we are not able to simulate every possible scenario so an algorithm has to self adapt to various unknown and unforseen situations.

Montana et al. [6] name their training sets "*poky1*" (slow nodes) and "*speedy1*" (fast nodes), while the test sets are called "*poky2*" and "*speedy2*". For clarification purpose we will name these "*slowTrain*", "*fastTrain*", "*slowTest*", and "*fastTest*". Three different parameter sets are trained on different test instances and then applied to the unknown instances that are left. In table Tab. V you can see the performance measured by the dropped packets of the parameter sets.

trained on	tested on			
	fastTrain	slowTrain	fastTest	slowTest
fastTrain	3.9%	N/A	3.9%	10.3%
slowTrain	N/A	0.3%	16.8%	2.0%
fastTrain+slowTrain	3.9%	0.9%	4.8%	0.6%

Tab. V: Dropped package rate of adapted parameter sets tested on different scenarios [6]

In the results it is clearly noticable, that parameter sets are performing better on statistically similar scenarios. For example a set developed on basis of *fastTrain* is performing well on *fastTrain* and *fastTest*, but its performance on *slowTest* is pretty poor. If the set was trained on a bigger variety of scenarios like the last one (trained on *fastTrain+slowTrain*) the performance is good through most of the test scenarios, if they are statistically similar. Another important fact is that the package loss is higher if the speed and therefore the range of geometrical configurations is bigger as you can see in Tab. V. This means that even an optimised algorithm does not perform perfectly if not a big amount of time and a very large amount of different scenarios are available for the training. Additionally Montana et al.[6] point out that they have reasonable expectations, based on own and others experience with optimizing parameter sets, that a performance degredation will always occur if one parameter set is applied to too many different scenarios or operating conditions.

As a conclusion there is to say that many different challenges in this topic have to be solved to make parameter optimisation with a *Genetic Algorithm* reasonable and exectuable in the real world. During this and the previous experiment we discovered some problems that have to be taken into account when it comes to generic adaption of such parameter sets. In the next section we will have a closer look at those and try to specify them more precisely.

IV. Problems and Challenges of generic network protocol optimisation

Finding a generic solution for parameter set optimisation in an *Organic Network Control System* would be a big step in computer science. The sheer amount of different protocols used in various networks with even more possible configurations seems to make this challenge nearly impossible to solve. But there are many approaches that have been proposed and allow further investigations of this topic. In the remainder of this article we want to point out some of the biggest issues that we will have to face if we want to build such a system successfully.

A. Choosing appropriate parameter sets

The first problem to face in a generic solution will be selecting an appropriate set of parameters for the current configuration of the network. Even if one node knows which type of network it is operating in and therefore which protocols are used, it needs to figure out the parameters of a protocol that are important for a better performance. Montana et al. [6] are pointing out, the parameter set is highly dependent of the current configuration of the network and influences the performance for the most part. Due to predefined parameter sets we were able to note a better performance after every cycle of optimisation in the experiments, but without pre-defining these parameters the system has to take a big foundational part of work on its own, detecting the correct parameters for adapting the system performance appropriately. For example in both experiments presented in this paper the parameters given to the *Evolutionary Algorithm* were completely different and so was its quality function, how the system can decide which ones are the ones to be taken into account always differs by the current use and configuration [6] of the protocol, e.g. in a *Mobile Ad-hoc* network download-bandwidth or dropped packets can be the main concern.

This problem is currently solved by the developers or engineers including a protocol to a simulation. As Tomforde et al. present in [8] to integrate a new protocol to their system some major tasks have been solved previously. The first task is to create a performance metric to be able to qualify good or bad performance. The second task is to describe the situation accompanied with a distance function to measure the difference between two situations. This includes defining the parameter that is most important for optimising certain processes, because it sets the focus of the optimiziation. The final task is to provide a simulation model for the simulation based testing on the Off-line optimisation component. A big problem with this approach is the big amount of different protocols used in the real world. It would take a lot of time to include all of them manually to the system. Therefore an automatic solution would be very desirable, but this needs further investigations and has not been proposed yet.

B. Resource and time management

Testing a new rule or individual is an important step when it comes to network protocols to avoid bad performance or

malfunction [3]. The question to face is in which extent the simulation should test a new developed parameter set. A more complex scenario would lead to a better quality of the developed parameter set, but also to more time spent. Tomforde et al. mention in their evaluation, that the time needed for a new generation of a parameter set highly depends on the complexety of the given test scenario. Montana et al. also point out that for a real life scenario of the Lakehurst dataset a simulation for one parameter set lasts about 7-8 minutes. "Since an optimisation run requires evaluation of 600-1000 different parameter sets, and hence the same number of executions of the simulation, an optimisation run takes multiple days on a single machine"[6]. Even if in the presented architecture time restrictions are not given it is not possible to estimate the amount of time that would be needed by an *Evolutionary Algorithm* to develop a new generation of parameter sets in a real world scenario, underlying the fact that a real world scenario will even be more complex than a simulated one.

The process is highly resource and time consuming [9] and for various networks it can not be realized for every node in a decentralized system as presented in [8]. These nodes can underly resource restrictions what makes running a simulation of that extent impossible.

Tomforde et al. are presenting various solution approaches and challenges due to this problem in [9]. To make the simulation faster the idea is to speed-up the start time of the *Evolutionary Algorithm* and get a faster algorithm to develop new generations. Therefore the quality of new individuals will decrease, but these can be approximated at runtime. Meeting the resource restrictions for some networks the architecture will have to be adapted to environmental properties. In a *Wireless Sensor Network*, for example, one node does not have the resources to handle a simulation of multiple hours. [9] Therefore it is presented that an "intelligent inter-extrapolation mechanism"[9] is needed. This means nodes should be able to communicate and update each other on current running services. Additionally the architecture needs to be half-centralized, which means having a node doing the simulation for other nodes in the network.

Hence, we see a big challenge in applying this architecture and simulation approach to the real world, but since real world scenarios are much more complex some challenges still have to be met especially considering time and resource mangement in networks like *Wireless Sensor Network* or *Mobile Ad-hoc*.

V. Summary

On the way to a solution for adapting network protocols generically some tasks still have to be solved to guarantee a optimal functionality in real world scenarios. We showed up an architectural approach, which can serve as a basis for further investigations due to its fast responding On-line adaption component and the Off-line rule generator. After explaininig the basic functionality and including components of this system the included experiments should lead to a better understanding of the range of scenarios where these systems can be used in. Nevertheless we pointed out the problem of selecting an appropriate parameter set due to these being needed for a proper adaption and highly depending on the current environment. Additionally it has to be dealt with time and resource management problems concerning the rule generation at Layer 2 (*Off-line optimisation*). To make this technology applicable to all types of networks solutions need to be found for resource restricted nodes as they exist in *Wireless Sensor Network* or *Mobile Ad-hoc* networks. The future task of investigation should be dealing with these problems as they need to be solved for a generic solution of ONC.

References

[1] Branke, J., Mnif, M., Müller-Schloer, C.: Organic Computing - Addressing Complexity by Controlled Self-organization, In: Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, pp. 185-191 (2006)

[2] Tomforde, S., Steffen, M., Hähner, J., Müller-Schloer, C.: Towards an Organic Network Control System, In: Proceedings of the 6th International Conference on Autonomic and Trusted Computing (ATC´09), Springer Verlag (2009) 2 - 15

[3] Tomforde, S., Hähner, J., Müller-Schloer, C.: Adaptive Control of Sensor Networks, In: Automatic and Trusted Computing, pp. 77-91 (2010)

[4] Web: The Network Simulator - NS/2, https://www.isi.edu/nsnam/ns/

[5] Eger, K., Hofeld, T., Binznehöfer, A., Kunzmann, G.: Efficient simulation of largescale p2p networks: Packet-level vs. flow-level simulations. In: 2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN 2007), Monterey Bay, USA, pp. 9-16 (2007)

[6] Montana, D., Redi, J.: Optimizing Parameters of a Mobile Ad Hoc Network Protocol with a Genetic Algorithm, In: GECCO '05 Proceedings of the 7th annual conference on Genetic and evolutionary computation, pp. 1993-1998 (2005)

[7] Web: Opnet Technologies, Inc. Wireless module, 2004. http://www.opnet.com/products/modules/wirelessmodule.html.

[8] Tomforde, S., Hurling, B., Hähner, J.: Dynamic control of mobile ad-hoc networks - network protocol parameter adaptation using organic network control. In: Proceedings of the 7th International Conference on Informatics in Control, Automation, and Robotics (ICINCO'10), (2010)

[9] Tomforde, S., Cakar, E., Hähner, J.: Dynamic Control of Network Protocols - A new vision for future self-organised networks. In: Proc. of the 6th Int. Conf. on Informatics in Control, Automation, and Robotics (ICINCO'09), pp. 285-290 (2009)

[10] Turgut, D., Daz, S., Elmasri, R., Turgut, B.: Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithmic approach. In: Proc. of the IEEE Global Telecommunications Conference (GLOBECOM 2002), pp. 62-66 (2002)

[11] Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., Schmeck, H.: Towards a generic observer/controller architecture for Organic Computing. In: Tagungsband der GI Jahrestagung, pp. 112-119 (2006)

[12] Müller-Schloer, C., Tomforde, S.: Organic Computing - Technical Systems for survival in the real world, Springer, pp. 6 (2017)