Cooperative Self-optimisation of Network Protocol Parameters at Runtime

Sven Tomforde[1], Jan Kantert[2], Sebastian von Mammen[1] and Jörg Hähner[1]
[1]*Organic Computing Group, University of Augsburg, Eichleitnerstr. 30, 86159, Augsburg, Germany*
[2]*SRA Group, Leibniz Universiät Hannover, Appelstr. 4, 30167, Hannover, Germany*

Keywords: Organic Computing, Intelligent System Control, Evolutionary Algorithms, Collaboration, End-to-End Communication.

Abstract: Network protocols are deployed in highly dynamic environments, but typically configured with a static setup of configurations. The Organic Network Control system (ONC) has been developed to alter protocol configurations at runtime. ONC is equipped with online learning capabilities and safety considerations. This paper presents a first TCP-based study on how this approach can be applied to end-to-end protocols and simultaneously alleviating the drawbacks of a simulation-based optimisation procedure. The paper explains the developed algorithm and demonstrates the benefit of the solution in an Omnet++ scenario.

1 INTRODUCTION

Large-scale data communication networks – such as the Internet – have experienced a dramatically increasing volume of data within the last decade. As a result, great efforts have been made towards improving the performance of the utilised protocols. Simultaneously, more efficient technologies have been investigated. One approach that researchers have followed is context-aware adaptation of protocols and their parameter configurations. The basic idea is to introduce "life-like" properties in communication – meaning that self-adaptation and learning components allow for highly robust and evolving solutions.

Previous work introduced the Organic Network Control (ONC) approach (Tomforde and Hähner, 2011), which adapts protocol parameter settings dynamically and without the need of human intervention. ONC works well for protocols with a local impact, e.g., mobile ad-hoc networks, which is caused by pure local knowledge and decisions. In order to shift the focus towards system-wide decisions, we introduce a novel concept to cooperatively optimise these parameters. To this end, we discuss a first TCP-based study and evaluate the benefits of the solutions.

The remainder of this paper is organised as follows. Section 2 describes ONC and related work. Afterwards, the novel cooperative approach to replace ONC's rule-generation component is introduced, exemplarily showing adaptation of TCP/IP parameters (Section 3). We discuss our experimental results in Section 4. Finally, Section 5 summarises the paper and gives an outlook to future work.

2 STATE OF THE ART

In this section, we first outline the workings of ONC. Second, we reference the amalgamation of its underlying mechanisms and inspirations.

2.1 Organic Network Control (ONC)

The goal of ONC is the integration of Organic Computing (OC) (Müller-Schloer, 2004) principles into existing network protocols, including self-organisation, self-optimisation, adaptivity, and robustness. ONC achieves this goal by automatic adaptation of network protocol parameters to environmental conditions. Thus, ONC strongly deviates from most established protocol implementations that maintain constant values for the respective parameters.

Figure 1 shows the layout of an OC architecture, i.e. the *Multi-Layer Observer/Controller architecture* (Tomforde et al., 2011b). In the context of the ONC system, the *System under Observation and Control* (SuOC) at Layer 0 represents one individual network protocol instance that runs on a particular host. Layer 1 contains an observer and a controller

Tomforde S., Kantert J., von Mammen S. and Hähner J..
Cooperative Self-optimisation of Network Protocol Parameters at Runtime.
DOI: 10.5220/0005520701230130
In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2015), pages 123-130
ISBN: 978-989-758-122-9

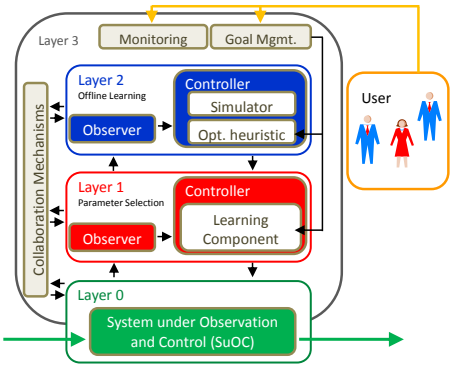

Figure 1: Multi-layered Observer-Controller Architecture.

component. The observer component is responsible for monitoring the SuOC's status and its environment and generates a situation description from the current observations. This is passed to the controller component. Based on this data, the controller decides whether and which changes should be introduced into the system. In addition, the current system description is used to reward any previous decisions' successes or to punish their failures, respectively. Making the right decisions and reinforcing the knowledge-base is typically realised by a modified Learning Classifier System, see (Prothmann et al., 2008). Due to safety and performance reasons, the online learning system is not allowed to generate novel behaviour – instead, it works on existing rules. In case of missing knowledge, i.e. if no decision can be made regarding the current situation, new rules are generated by Layer 2. In particular, it finds the best possible action for a given situation by means of a simulator and an according optimisation component. Finally, Layer 3 provides an interface to the user and to neighbouring systems.

ONC has been applied to different types of network protocols, including reliable broadcast algorithms for mobile ad-hoc networks (Tomforde et al., 2011a), mode-selection protocols for wireless sensor networks (Tomforde et al., 2010), and Peer-to-Peer protocols (Tomforde and Hähner, 2011).

2.2 Related Work

ONC is a representative of **decentralised approaches** to adapt network protocols. Several related approaches can be found in the literature that are specific to certain protocol implementations, whereas ONC is applicable to network protocols with variable parameters in general. For instance, Su et al. describe a mobility-adaptive self-parametrisation of different unicast and multicast routing protocols in the MANet-domain – depending on a positioning system. Boleng determines mobility by locally analysing the observed neighbourhood. Considering the average link duration as mobility metric, he uses the received values to control data routing in the so-called *Adaptive Location Aided Routing from Mines* protocol. Finally, Stanze et al. describe a system for mobility-adaptive self-parametrisation of a routing protocol in MANets. To this end, they measure mobility by using the *MANET Relative Velocity Indicator* protocol. These approaches are only feasible in the MANet-domain as they work directly on the mobility aspect.

Further concepts for self-configuration are already **considered in the design of protocols**. Examples are the *congestion avoidance* mechanisms in TCP (Martin et al., 2003) and the *collision detection* for Ethernet-protocols (Kleinrock and A.Tobagi, 1975). Considering adaptivity aspects within the protocol logic has drawbacks: The logic has to be changed and a cooperation with standard non-adaptable protocols might not be possible (besides further aspects such as limited re-usability, limitation to only one purpose). Again, this results in problem- and protocol-specific solutions without the possibility of a generalised approach. The self-configuring approaches outlined above have in common that individual network nodes can adapt based on locally available information. In contrast to such decentralised approaches, systems have been investigated where one **central, network-wide instance** concerts parameter changes. The most prominent examples have been presented by Ye et al. (Ye and Kalyanaraman, 2004) and Georganopoulos and Lewis (Georganopoulos and Lewis, 2007). Closely related is the concept of **protocol stack composition**, which aims at providing a more general solution to adapting protocol parameters. Instead of interfering with the protocol logic, protocol stack composition allows to switch between protocols and stacks dynamically (Rosa et al., 1997). Several different systems have been developed – popular representatives are *Appia* (Miranda et al., 2001) and *Horus* (van Renesse et al., 1996). Here, protocols with static settings are selected on demand from a predefined set of solutions without any chance to extend it at runtime. Furthermore, they rely on a centralised element to observe and, if need be, reconfigure the protocol stack for the whole network. Due to centralisation, challenges arise such as additional bandwidth usage and a single point of failure.

3 COOPERATIVE PARAMETER OPTIMISATION

The ONC approach as introduced in Section 2.1 works well for protocols with a local impact. In

particular, this means that altering protocol parameters has mainly a local focus, which is e.g. the case for MANet protocols. Besides the potential benefit demonstrated in previous work (see e.g. (Tomforde and Hähner, 2011; Tomforde et al., 2010)), the concept has drawbacks that restrict its applicability. For instance, we rely on the existence of a simulation model that a) approximates reality very accurate and b) models the behaviour independently of all other nodes in the network. Especially for protocol algorithms that have more than just a local influence, the simulation-based rule-generation approach is limited in finding the desired behaviour. One example of previous work, where such a drawback has been observed is the control of the Peer-to-Peer client *BitTorrent* (Tomforde and Hähner, 2011). Besides these concept-inherent limitations, processing time is another drawback. Simulation-based optimisation approaches are time-consuming.

In general, there are two different approaches to counter these limitations. The first is the *pure machine learning* idea: Neglecting safety considerations, one could deactivate the Layer 2-based rule generation and equip Layer 1 with an exploration mechanisms. In particular, the Learning Classifier System of Layer 1's controller could be implemented as a standard *eXtended Classifier System* (XCS) (Wilson, 1995). Besides the safety-aspects, this approach is problematic due to the extremely long convergence durations. Additionally, bad protocol parameter settings (i.e. a result of machine learning) affect the communication ability in such a way that the system might get trapped in a non-communicating state. As a result of these consideration, such a solution is not feasible.

The alternative is to replace the simulation-based solution by other mechanisms. Since simple concepts such as approximation or computation of optimal settings are not available for network parameter estimation, we replaced the simulation-based rule-generation by a context-dependent cooperative solution. The idea is to get rid of the simulation model and try possible behaviour directly in the real system – but still consider safety.

3.1 Modification of the ONC Architecture

In general, there is significant work available on optimising TCP parameters and techniques. For instance, Nichols and Jacobsen summarise a set of solutions for most of the current TCP problems (Nichols and Jacobson, 2014). Furthermore, Dukkipati et al. already presented a solution for TCP's "slow-start" problem, which is also addressed by our approach (Dukkipati et al., 2010). Within this paper, TCP serves as an example for connection-oriented protocols and consequently as use case for out experiments. The general idea is transferable to similar protocols – we therefore neglect a deeper comparison with comparable TCP modifications.

Layer 0: In order to be able to adapt the TCP/IP host to changing conditions, Layer 0 needs the possibility to access variable parameters and to monitor the current situations. According interfaces have been implemented. Details on implementation parameters are given in the context of the simulator (see Section 3.3).

Layer 1: The parameter configuration loop of Layer 1 consists of an observer and a controller component. The observer derives a situation description from the SuOC's accessible attributes. In the TCP/IP setting, we use the following situation description:

1. The (averaged) duration of a TCP connection from or to this host.
2. The (averaged) number of TCP connections that are open simultaneously.
3. The (averaged) round-trip time per packet.
4. The particular physical medium used as basis for the communication (e.g. WiFi or Ethernet).
5. The maximum transmission rate for the current connection (e.g. $100 Mbit/s$).

Based on this situation description, the controller performs two tasks: a) *evaluate the success* of the last action and b) *choose the next action*. We implemented Layer 1's controller as a modified variant of Wilson's eXtended Classifier System (Wilson, 1995) (see (Prothmann et al., 2008) for details). We measure the system's performance as the throughput without overhead ("goodput"). I.e., we measure the round-trip time of packets and estimate the throughput based on the packet size and the considered period. From this accumulated value we subtract the overhead.

In order to have a normalised reference value, we calculate the round-trip time as given by Equation 1. Thereby, the abbreviation RTT specifies the (averaged) round-trip time – the index c refers to the currently observed one, *max* to the maximal observed one, *min* to the minimal observed, and n to the (finally) normalised value. As a result of this normalisation, RTT_n is within the interval $[0,1]$. In addition, the actual value within this range is adapted in relation to the possible value for maximum and minimum.

$$RTT_n = 1 - \frac{(RTT_c - RTT_{min})}{(RTT_{max} - RTT_{min})} \quad (1)$$

Layer 2: The most important modifications to ONC are concerned with the rule-generation component. Again, this component consists of an observer and a controller part. In case of periods of inactivity or low resource utilisation, parts of the bandwidth can

be used to cooperatively optimise parameter settings. This is the task of the observer-controller tandem on Layer 2. In contrast to the previous architecture as depicted in Figure 1, no simulation tool is available nor necessary, and it is therefore replaced by a component that is responsible for: a) finding *cooperation partners*, b) generating (test) traffic to utilise the bandwidth, and c) cooperatively optimising the parameter settings for both ends of the TCP connection using a heuristic. For part c), the same optimisation approach as used for ONC's Layer 2 (see Section 2.1) can be applied. The other two aspect are described in detail in the next subsection.

Layer 3: The third layer of the architecture encapsulates the collaboration and cooperation with other entities, as well as with the user. It maintains and updates a list about available neighbours. This list serves as set of potential cooperation partners. Details are given in the remainder of this section.

3.2 Cooperative Self-optimisation of Protocol Parameters

Our proposed self-optimisation approach consists of two parts: a) identify *cooperation partners* and establish a test-connection with adaptable TCP parameters (this assumes that all hosts run the same TCP implementation and access the same parameters using the same names), and b) find the best setting for the specified conditions.

a) Cooperation Partners: Hosts are assumed to be aware of other hosts within their vicinities – e.g. due to observing addresses in TCP packets transmitted over the physical medium. Hence, each host maintains a list of potential cooperation partners. Candidate partners have to be selected from this set. Thereby, different strategies are possible, e.g.: a) a random selection, b) the last communication partners of TCP connections established by or to the specific host, c) hosts that will most probably be the next communication partners, and d) hosts that exhibited bad/good RTTs in the past. We decide to keep the approach as simple as possible and therefore choose random selection. In addition, two restrictions have to be considered: 1) the candidate partner does not run an ONC module and 2) the candidate partner refuses to cooperate (e.g. due to over-utilisation). Both restrictions are covered by introducing a time-out and then starting the selection process again. Those not answering are further put on a blacklist.

b) Optimisation Process: After selecting partners, the cooperative testing of parameter settings is initialised. We distinguish between an *active* and a *passive* role: The requesting host is active and therefore responsible to generate candidate parameter settings and set up test traffic. The other cooperation partner is passive and applies the requested parameters for a given time interval. During this period, it accepts the incoming TCP connections of the active host, accepts the data (and deletes it), and observes the situation. Within such an evaluation cycle, one parameter set can be evaluated by the active host. This loop – testing a variety of candidate parameter sets – is processed until a stop criterion is reached (i.e. number of maximum cycles or no improvement for a specific number of cycles). In order to generate candidate parameter sets to be tested, we used an Evolutionary Algorithm working on a bit string representation. As test data, we use a randomised html-traffic generator as available in the simulator and configure it with the requested conditions (i.e. number of open TCP connections, available bandwidth, etc).

The bit string encoding the candidate solution consists of the 10 parameters listed in Table 1. Thereby, the ordering in the string reflects the ordering in the table and the number of bits used per parameter is given in the third column. Based on this string, the genetic operators I) *selection*, II) *crossover*, and III) *mutation* are applied. For I), we use a probability-based selection of candidates within the population (meaning from the set of known rules). The probability of becoming a parent becomes higher with increasing performance. The second aspect – cross-over – decides how the new candidate is generated: either by combining the two selected parents or randomly. Typically, EAs are configured with a high cross-over probability (meaning to utilise selection). Therefore, we set this parameter to 0.95. The operator splits both bit strings of the parents at one (randomly chosen) point and recombines them while switching the ends (one-point-cross-over). Alternatives include variants of two- and more-point cross-over. Finally, additional randomness comes in with the mutation parameter (III). Mutation introduces more exploration capabilities, but high probabilities will mislead the process towards random search behaviour. Hence, the mutation probability is set to a low value (i.e. 0.1).

c) Optimisation Protocol: Standardised TCP messages specifying the request for cooperation, the acknowledgement/reject, and the transmission of the candidate parameter sets have been defined to allow the possible partners to understand each other.

3.3 Implementation in Omnet

Each host in Omnet (i.e. its TCP instance) serves as SuOC according to the ONC architecture. Upon each of these SuOCs, ONC is set up according to

the modified architecture (see Section 3.1). We used the Omnet++ network simulator (Varga, 2001) as basis for the development and evaluation. Omnet++ is a standard tool in network protocol simulation and has been widely used in academia and industry. We implemented the SuOC as instance of the StandardHost module in the INET-Framework (INET Project, 2014) of Omnet++. This module realises the TCP/IP protocol stack and can easily be connected to a network within the simulator. The sub-modules containing the particular functionality of the TCP and IP layers are configured at simulation start-up using NED-parameters (i.e. special variables belonging to one module; NED is Omnet's control language). Initially, these parameters are set using standard configurations, but they can be altered individually – which is important to simulate the desired behaviour.

Table 1: TCP parameters, standard configuration (INET), and number of bits for encoding in the optimisation process.

Parameter	*Value*	*Bits*
mss	536	10bits
advertisedWindow	$14 \times mss$	5bit
tcpAlgorithmClass	TCPReno	2bit
delayedAcksEnabled	false	1bit
nagleEnabled	true	1bit
limitedTransmitEnabled	false	1bit
increasedIWEnabled	false	1bit
sackSupport	false	1bit
windowScalingSupport	false	1bit
timestampSupport	false	1bit

Table 1 lists the available TCP parameters of the StandardHost module. Thereby, *mss* refers to the *maximum segment size* (see RFC 793), *advertisedWindow* refers to the maximum size of the buffer for incoming messages, *tcpAlgorithmClass* is used to select the desired TCP variant (TCPReno, TCPNewReno, TCPTahoe, TCPNoCongestionControl, or DumpTCP), *delayedAcksEnabled* enables or disables the delayed ACK-algorithm (see RFC 1122), *nagleEnabled* chooses to use the Nagle-algorithm (see RFC 896), *limitedTransmitEnabled* activates the Limited-Transmit-algorithm (available for TCPReno, TCPTahoe, TCPNewReno, and TCPNoCongestionControl, see RFC 3042), *increasedIWEnabled* controls the *Increased Initial Window* (see RFC 3390), *sackSupport* activates the support for *selective acknowledgments* (is automatically used if both endpoints support this option, see RFCs 2018, 2883, and 3517), *windowScalingSupport* activates the support for *window scaling* (is automatically used if both endpoints support this option, see RFC 1323), and finally *timestampSupport* activates the support for timestamps (see RFC 1323). We decided to use all of these parameters and not just the set of most promising ones, since we wanted the system to automatically figure out the impact of each particular parameter.

4 EVALUATION

This section evaluates the potential benefit of the previously explained approach. Therefore, simulations within the network simulator Omnet++ (Varga, 2001) have been performed.

4.1 Experimental Setup

In order to simulate the cooperative parameter optimisation mechanism, we implemented the approach as described in Section 3.3 directly in Omnet by defining new modules. Afterwards, we set up a test network that consists of two hosts (*cli*0 and *cli*1) running the ONC approach and communicating with each other. A third component (*srv*) provides content and does not understand the ONC messages. We also tested the approach in simulations of larger networks (i.e. 5, 10, and 20 hosts). Since the results are comparable to those of this simple setting, we neglect this information in the context of this paper.

4.2 Experimental Results

We start the evaluation with an investigation of the underlying fitness landscape for the parameter space. Afterwards, the learning behaviour at runtime of the developed solution is analysed. This is done in comparison to the usage of standard TCP parameters available in Omnet++ (which are already a near-to-optimal configuration, see fitness landscape). Finally, the individual results for exemplary simulation runs are generalised by deriving averaged values over different scenarios with varying conditions.

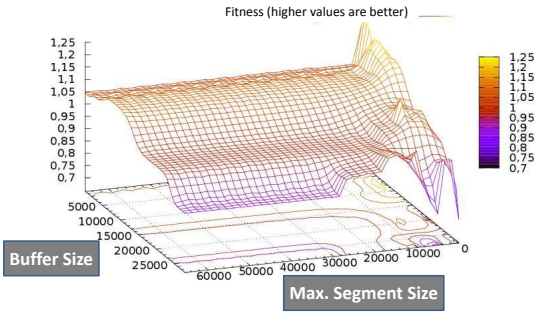

Figure 2: Fitness Landscape.

4.2.1 Fitness Landscape

Initially, we analysed the underlying fitness landscape for the optimisation problem. For better illustration, the following section focuses on the two most important parameters: (a) *maximum segment size* and (b) *buffer size*). We set up a simulation in Omnet++ without *ONC* module – it contains just the simulated client and server components. The simulation covers one complete day (i.e. 24 hours). As a result, Figure 2 visualises the relation between the two parameters: smaller values for the maximum segment size and higher ones for the buffer size are better. In general, the fitness landscape describes an almost flat and smooth surface – meaning that the optimisation is a minor complex problem. Taking further parameters into account makes the optimisation problem more complex. Especially the true/false decisions for enabling or disabling certain functionalities results in hard breaks in the landscape (difficult to optimise).

4.2.2 Online Learning Behaviour

In order to simulate the online learning behaviour and the success over time, we simulated five consecutive days in Omnet. Both clients randomly access websites at the server component. One of the clients (we refer to this as "passive" – i.e. *cli1*) keeps its standard TCP configuration during operation – this means that it does not activate its ONC-based parameter adaptation mechanism during normal operation. This is only done in case of incoming requests for cooperative optimisation by the other client.

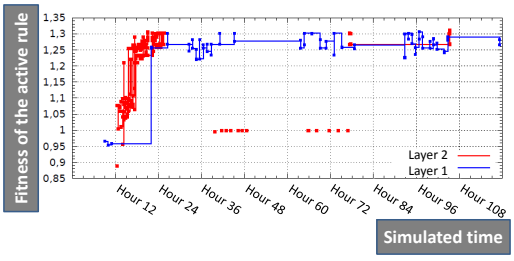

Figure 3: Change of the fitness of the active rule over time; starting with a bad parameter setting.

The active client (i.e. *cli0*) simultaneously executes the ONC component – it evaluates its behaviour and adapts the parameter settings using ONC. Thereby, it starts the cooperative optimisation in case of missing knowledge or unsatisfying performance. Initially, the active client starts with an empty rule base – meaning there is no prior knowledge or any information about the desired parameter settings.

The goal for the active client is to find well-performing parameter settings – without any initial information. As starting configuration we chose bad settings according to the fitness landscape characterisation depicted in Figure 2 – in particular, we identified one of the "deepest" valley (i.e. $mss = 40,000$ and $buffer = 20,000$).

Figure 3 illustrates the achieved results for the active client. Thereby, the figure distinguishes between Layer 1 (blue line) and Layer 2 (red line). The values have been accessed at the begin of each new parameter set being active and at the end. From this figure it is visible that the start parameter configuration is far from being optimal, values for the fitness between 0.9 and 1.0 can be observed. The theoretic optimum in this scenario (which depends e.g. on the channel characteristics) is $\sqrt{2}$, meaning about 1.414. The performance of Layer 2 quickly approaches values between 1.0 and 1.2 that further improve to values between 1.2 and 1.3 during the simulation.

The randomised traffic generation pattern resulted in the highest load within the first 12 hours of the first day. Afterwards, the ONC component of the active client recognises that no traffic is observed (i.e. there is the possibility to find new parameter settings) and the current setting seems to be non-optimal (i.e. far from the theoretic optimum). In the course of the simulation, eventually new traffic comes in and the ONC system proceeds in adapting the parameter sets. This results in a continuously increasing performance of Layer 1. At the end of the simulation (i.e. at the end of day 5), the performance is constantly near to 1.3 (this is about 92 % of the optimum).

For comparison reasons, we performed the same simulation without the ONC component. This means that each participant performs the standard TCP parameter set as listed in Table 1. This standard parameter set represents the hand-optimised solution available in the simulator – in general, this should be the setting that works best *on average* in each possible situation. The results are illustrated by Figure 4. Comparing both figures, it can be observed that the ONC approach finds similar parameter settings as the standard parameter set (i.e. the last parameter sets are: $mss = 586$ and $buffer = 7,504$) – but without the need of prior knowledge, without manual optimisation, and with the potential benefit to adapt automatically to changing conditions and different physical connections. This is exactly what we wanted to achieve with the developed concept.

Another observation that can be made in this single-run scenario is that the performance is changing - it is not just a continuous improvement. In contrast, periods of decreasing performance can be observed. This is due to the learning mechanism that follows a roulette-wheel approach considering the fit-

ness of the available rules. Please note that the results shown in the figures reflect just *one* simulation run and have not been averaged over several runs. This is due to demonstrating the specific behaviour – the effects of individual rule adaptations are not visible in aggregated figures based on several runs. Averaged values are considered in the following part.

4.2.3 Evaluation of Multiple Scenarios

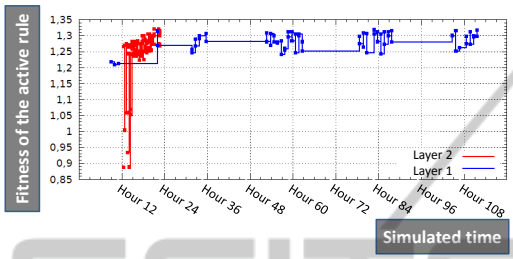

Figure 4: Change of the fitness when performing the standard parameter sets.

The third part of the evaluation considers the performance of ONC in a more generalised way. Thereby, we take 27 runs of the simulation with varying seeds into account and derive the statistical results. The considered simulation period has been increased to 15 days to get an impression about the long-term behaviour of the system. In order to be able to compare the results of all runs, we deactivated the dynamic adaptation of the RTT estimation. This is done by defining static RTT_{min} (here: $0,013$ ms) and RTT_{max} (here: $4,3$ ms) values (cf. Equation 1). Figure 5 illustrates the results for the ONC solution. In comparison, Figure 6 illustrates the behaviour in case of using just the standard parameter settings (i.e. deactivated rule-adaptation for all hosts). The figures display the average during the simulation time (blue line), the corridor within that values are measured (i.e. minimum and maximum values over time), and the standard deviation.

Similar to the previous experiments, ONC (i.e. client0) starts with a bad parameter setting (i.e. again: $mss = 40,000$ and $buffer = 20,000$). The figure shows that the effect observed for the single run can be generalised for several runs. Thereby, we depict the corridor of the observed behaviour using the minimum and maximum values at the certain point in time for the system performance. The most obvious insight by considering this corridor is that its size decreases over time for Layer 1; in contrast, the behaviour does not change significantly for Layer 2. This is exactly what the learning approach is assumed to achieve: With increasing number of classifiers and growing experience with these, the system gets aware of the particular impact (i.e. it actually *learns* the desired behaviour). Layer 2 tries to find novel behaviour for previously untested situations – therefore, no recognition of patterns is possible (i.e. it has to discover the fitness landscape again for each new task).

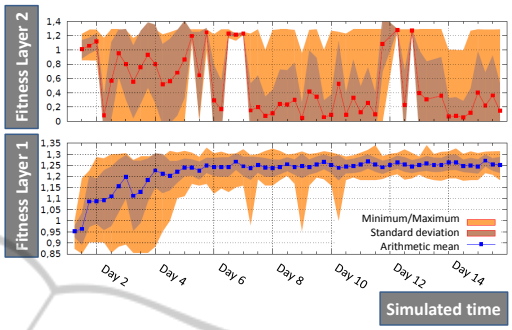

Figure 5: Performance of Layer 1 and Layer 2 for 15 simulated days (start = bad parameter settings).

A second major observation can be made when comparing Figure 5 and Figure 6. For the ONC-based variant, the learning behaviour is visible (decreasing corridor size) – while it stays static for the non-ONC version. This reflects the expected behaviour, since the non-ONC version performs the standard parameter sets without modifications and hence results in a statistically static behaviour by design.

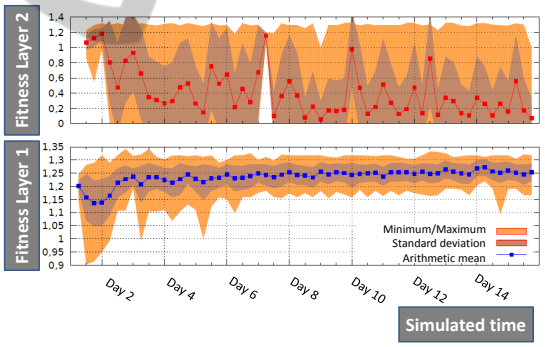

Figure 6: Performance of Layer 1 and Layer 2 for 15 simulated days (standard parameter settings).

4.3 Discussion of the Achieved Results

The simulation results demonstrate the potential benefit of utilising the proposed cooperative self-optimisation to find the best possible parameter settings in connection-oriented network protocols. The scenario is a perfect example to demonstrate OC's vision of moving design-time decisions to runtime and into the responsibility of the system. Instead of spending resource- and time-consuming optimisation efforts at design-time to find the best possible parameter settings – mostly those configurations are requested that work best *on average* for all foreseen situations

– the system itself finds the best parameter settings in each situation. This results in a highly self-adaptive and robust behaving system which can also adapt to unforeseen situations.

Since a local perspective is not enough for connection-oriented protocols to either set consistent parameters or to measure the success, cooperative solutions are needed. This matches again perfectly with OC's research agenda. In this paper, we developed a self-organised solution which cooperatively finds the best strategies at runtime. The effect of testing and changing parameters always appears for two partners – thereby, we created a self-organised way of changing the structure of the system (i.e. which entity is cooperating with which other entity). From a performance perspective, the developed solution is able to find parameters that show the same performance as the optimised standard parameters in undisturbed situations. However, the system is expected to find superior parameter settings in disturbed situations (i.e. very high latency or very high packet loss).

5 CONCLUSIONS

This paper presented a novel distributed approach for self-optimisation of data communication protocol parameters. Based on previous work in the context of the Organic Network Control system (ONC), we explained a cooperative approach to find and test parameter settings for TCP. The approach has been evaluated in a Omnet++-based simulation and demonstrated the potential benefit. In contrast to other solutions from the state of the art, our method works without prior knowledge, considers safety-boundaries, and self-improves its behaviour over time.

Current and future work focus on further improving the mechanism and applying it to continuously changing conditions. In the scenarios considered in this paper, we demonstrated that the solution is able to find similar parameter settings as those initially available in the TCP implementation – but without any prior knowledge and without the need of time-consuming optimisation processes at design-time. We only simulated slightly changing conditions that do not affect the physical medium (i.e. we only work on Ethernet connections). Currently, we analyse how the behaviour changes in case of replacing the physical medium (e.g. Ethernet vs. WiFi) during operation.

REFERENCES

Dukkipati, N. et al. (2010). An Argument for Increasing TCP's Initial Congestion Window. *ACM SIGCOMM Computer Communications Review*, 40:27–33.

Georganopoulos, N. and Lewis, T. (2007). A Framework for Dynamic Link and Network Layer Protocol Optimisation. *Proc. of Mobile and Wireless Communications Summit*, pages 1–5.

INET Project (2014). The INET Framework. http://inet.omnetpp.org/.

Kleinrock, L. and A.Tobagi, F. (1975). Packet Switching in Radio Channels: CSMA Modes and Their Throughput-Delay Characteristics. *IEEE Trans. on Com.*, 23(12):1400–1416.

Martin, J., Nilsson, A., and Rhee, I. (2003). Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, 11(3):356 – 369.

Miranda, H., Pinto, A., and Rodrigues, L. (2001). Appia: A Flexible Protocol Kernel Supporting Multiple Coordinated Channels. In *Proc. of ICDCS '01*, pages 707 – 710. IEEE.

Müller-Schloer, C. (2004). Organic Computing: On the Feasibility of Controlled Emergence. In *Proc. of CODES and ISSS'04*, pages 2–5.

Nichols, K. and Jacobson, V. (2014). Controlled Delay Active Queue Management draft-nichols-tsvwg-codel-02. Technical report, Pollere Inc. and Google.

Prothmann, H., Rochner, F., Tomforde, S., Branke, J., Müller-Schloer, C., and Schmeck, H. (2008). Organic Control of Traffic Lights. In *Proc. of ATC-08*, volume 5060 of *LNCS*, pages 219–233. Springer Verlag.

Rosa, L., Lopes, A., and Rodrigues, L. (1997). Appia to R-Appia: Refactoring a Protocol Composition Framework for Dynamic Reconfiguration. Technical Report 1, Univ. of Lisbon, Dep. of Informatics.

Tomforde, S. and Hähner, J. (2011). *Biologically Inspired Networking and Sensing: Algorithms and Architectures*, chapter Organic Network Control, pages 11–35.

Tomforde, S., Hurling, B., and Hähner, J. (2011a). Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks. In *Informatics in Control, Automation and Robotics*, pages 91 – 104. Springer.

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. (2011b). Observation and Control of Organic Systems. In *Organic Computing - A Paradigm Shift for Complex Systems*, pages 325 – 338. Birkhäuser.

Tomforde, S., Zgeras, I., Hähner, J., and Müller-Schloer, C. (2010). Adaptive Control of Wireless Sensor Networks. In *Proc. of ATC'10*, pages 77 – 91.

van Renesse, R., Birman, K. P., and Maffeis, S. (1996). Horus: a flexible group communication system. *Communications of the ACM*, 39(4):76 – 83.

Varga, A. (2001). The OMNET++ discrete event simulation system. In *Proc. of European Simulation Multiconference*, pages 319–324.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175.

Ye, T. and Kalyanaraman, S. (2004). A recursive random search algorithm for network parameter optimization. *SIGMETRICS Perform. Eval. Rev.*, 32(3):44–53.