Towards User-centric Network Optimization Engine

Elena Meshkova, Andreas Achtzehn, Janne Riihijärvi and Petri Mähönen
Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {eme, aac, jar, pma}@mobnets.rwth-aachen.de

Abstract—The increasing diversity of applications as well as the introduction of new wireless last-mile techniques imposes additional challenges on both terminals and entire networks. A user- and application-centric approach to networking is expected result in better user satisfaction and allow for new market opportunities for network infrastructure providers. However, this architectural view leads to increased network complexity and, therefore, requires new autonomic network management and optimization techniques. In this article we focus on one aspect of this domain: network protocol stack optimization for wireless access networks. We provide a design of an autonomic engine that aims to select and adjust parameters of the communication stack in order to achieve high user satisfaction using minimal reconfiguration time. It relies on two basic constructs: utility-based performance assessment, and metaheuristic algorithms for fast optimization. We demonstrate that such an engine allows for efficient autonomous switching between different wireless access methods and consequently significantly improves the performance of the user device using cross-layer parameter optimization.

I. INTRODUCTION

The complexity of modern computer networks is rapidly increasing. Popular applications like YouTube, P2P file sharing, video- and audio-streaming inflict different, often contradictory, requirements to the access infrastructure. Wireless access technologies further contribute to the complexity as they perform suboptimal with the default settings of long-established network protocol stacks [1]. In times of diminishing profit margins, economical considerations need to be carefully regarded not only at the stage of the network system design, but also at run-time. Therefore, the work of network designers and operators is becoming difficult since a considerable amount of knowledge and on-hand experience is needed to create user-centric heterogeneous networks. The need for additional tools for network management and optimization is raising.

We propose an architecture and provide first performance results of a cognitive engine that aims at partial automation of network design and its run-time optimization. We approach the problem from a network protocol stack configuration viewpoint. This is a relevant task since any application on an end-user terminal requires various network protocols to work optimally together. Additionally each protocol can be fine-tuned by adjusting parameters, thus affecting all other protocols in the stack. Examples of parameters are timeouts or maximum payload sizes. The final goal is to obtain the best combination of protocols and their parameters for a certain network configuration to maximize customer and operator satisfaction. Potentially a large number of interrelated parameters have to be changed simultaneously, which results in a NP-hard combinatorial optimization problem.

In this paper we limit the description of our tool to its application to run-time optimization of individual terminal devices in the presence of various wireless access networks. However it is not designed solely for this purpose. The optimization can be carried out at various levels of granularity: at network, protocol stack of an individual node, and software components scale. It also can be conducted on static, long-term configuration base, and in a dynamic run-time fashion. The areas of applications of the engine are networks that are likely to experience bottlenecks and suboptimal performance with a standard configuration base, and in a dynamic run-time fashion. Typical examples are wireless sensor networks, cognitive wireless networks and configuration of Wi-Fi networks.

We show that the cognitive engine is capable of simultaneous adjustment of network protocol parameters, as well as of switching between various network access methods, achieving near-optimum network performance at the price of sufficiently short reconfiguration time. On average it shows better results than a user who manually switches between network access methods, leaving other settings unchanged at the default state.

In Section II we describe the overall architectural design of the auto-configuration engine. We present the results obtained with an experimental implementation of the cognitive engine in Section III. We demonstrate the benefits that emerge when cross-layer parameter optimization is performed in such scenarios. We conclude the paper in Section IV.
The overall architecture of the proposed cognitive engine is given in Figure 1. It is based on the specification of the autonomic control loop by the research group of IBM [2] and resembles the definition of intelligent agents commonly used in the AI community [3]. The main idea is that the optimizer considers the networks as a “black box” with a collection of numerous known components, each of them having several adjustable parameters (often referred to as “knobs” or “actuators”). However, no prior knowledge on the behavior of these components is required. On a high abstraction level one may think of parameters as variables in a software component. Optimization targets are given by application-specific utility functions which depend on performance metrics, regarded as attributes (“dials” or “sensors”). Examples of parameters are the maximum number of retransmissions in MAC protocols, contention window sizes or the flavor of TCP used. Common examples of attributes are network delay, throughput or power consumption. The adjustment of knobs leads to changes in readings of the dials. If we achieve the optimal setting of knobs values, we will maximize values of relevant dials and therefore reach the utmost user satisfaction.

Utility functions are the most flexible way to evaluate system performance, as they allow for a quantifiable expression of user needs and can be used by autonomic systems [4], [2]. They are known to be difficult to design, but embody most of the objectives in networking used today such as QoS classes or logarithmic utility for TCP. Utility functions allow us to define complex criteria for user satisfaction and network performance (Figure 2). They can be specified either as analytical functions or as sets of logical constructs, for example via if statements. The utility depends on attributes, that in turn take parameters and other constraints into consideration. They might also incorporate economical metrics like charging fees. Further formal specification of the utility dependencies, as well as specification of the optimized entity is given in [5].

The platform initially requires inputs from the user and the component specifications. The former contains the description of the imposed policy constraints, including required network functionalities, and the expected utility. The latter specifies the protocols and the related components that are considered for the optimization. Prior to protocol stack parameter adjustment we can reason on dynamic composition of network services in order to obtain suitable protocol stacks [6]. The framework must also incorporate a knowledge base that stores the above information along with network performance measurements.

The configuration engine acquires self-adaptive features through a metaheuristic algorithm, namely simulated annealing [7]. This method enables fast convergence to the stable near-optimum solution in the parameter optimization space. It is also known to be effective for “black box”-type of problems. It has been shown to have good asymptotic convergence properties. Besides, this method potentially allows the use of a smaller number of parameter sets, compared to, for example, genetic algorithms [3]. We define the search space for simulated annealing as a multi-dimensional landscape where each dimension corresponds to one adjustable parameter. Each parameter might take a number of predefined states or values. In this paper we regard our optimization problem as utility maximization task in this search space.

The cognitive engine tries to conduct optimization in an iterative manner. The first optimization cycle is initiated immediately at the start of the application, after acquiring the initial state of the monitored entity (in our case it is a single network node). This search phase is typically the longest, as no prior information on the network is available. Every search cycle a new parameter value set is evaluated and the resulting utility is obtained. The search stops either when the system cools down, according to the classical SA, or when the utility is not further improved for a certain number of iterations and it is close to the user-defined maximum. The cooling scheduler and the utility threshold allow for a performance trade-off between fast convergence of the optimization process and the quality of the results achieved. The later enhancement is also required to limit the search phase that is likely to cause a high variation in the system’s performance. Simulated annealing optimization is re-initiated in case of performance degradation, i.e. if the utility drops. This process is also repeated regularly for short times (reheating in SA terms) in order to better explore the local search space in a hope to obtain a better utility or detect a change in network conditions.

Fig. 2. Utility as function of two attributes: goodput and cost.

![Utility as function of two attributes: goodput and cost.](image)

### III. Results

We have tested the performance of the cognitive engine with different scenarios in various environments including sensor networks [8]. In this paper as a representative example we have chosen a scenario where a user runs a YouTube-type of application. It is specified in the Qualnet network simulator [9]. The engine itself is realized using MATLAB. It queries the simulator every N milliseconds and gets statistical information on the performance of the nodes of interest. Using the same interface, parameters of the nodes are adjusted as result of the
The performance of the high throughput network severely decreases if the user experiences too high delays, so the utility is zero. At the other end, if the expected goodput is less than 1.5 Mbps the user leaves the network as she is less satisfied. If the expected goodput stays constant through the download, the longer the download time the less satisfied the user becomes. If the expected goodput is less than 1.5 Mbps the user leaves the network as she experiences too high delays, so the utility is zero. At the other extreme fast downloads with goodput of more than 7 Mbps result in very high satisfaction. The cost function models a goodput-dependent price plan for network usage. The higher the used data rates, the more a user need to pay. Per-data rate price plans are advantageous to future network providers as they allow for flexible charging, better use of existing network capacity and therefore higher return of investment.

In Figure 3 simulation results are provided. In the ideal case the user has to switch the interface only once when the performance of the high throughput network severely decreases (saturation point). However, high goodput does not always lead to high utility. We hereby prove an initial assumption that the more realistic and complex utility metric is beneficial for both providers and users, as provider does not loose its customers due to congestion, and users have their primary interests satisfied. The use of advanced utility and protocol stack optimization capabilities led to an average performance increase of 30% in this scenario compared to the manual switching between access points conducted by the user with the default parameter settings. Overall performance improved as the initial learning phase was followed by a phase of high utility due to good parameter settings. Learning effects are also visible in the next probing period where more parameter settings were evaluated as the result of worsening network conditions. After reaching the saturation point, the cognitive engine was able to quickly switch the access network and choose new settings that kept utility above the level obtained by using default settings.

IV. Conclusions

In this paper, we have described the design of an autonomic user-centric network optimization engine for wireless access systems. We have discussed how utility function descriptions and metaheuristics-based optimization can lead to an efficient configuration framework. The initial results shown by the developed cognitive engine are very promising.

We are working on enhancements to the engine with learning capabilities using probabilistic graphical models. We also plan to publish the reference implementation of the prototype and increase the number of its interfaces to interact with other network simulators and software defined radio platforms.

ACKNOWLEDGMENTS

Authors would like to thank RWTH Aachen University and EU (Aragorn-project) for financial support.

REFERENCES


Fig. 3. The time-evaluation of performance and utility in YouTube scenario showing adaptive capability of the system and optimization gains. Figure shows smoothed averages from a large number of simulations, some of the raw data points are depicted on the background to show sample variance.