Enabling Run-Time Utility-Based Optimization through Generic Interfaces in Wireless Networks

Krisakorn Rerkrai, Jad Nasreddine, Janne Riihijärvi, and Petri Mähönen
Institute for Networked Systems, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email:{kre,jad,jar,pma}@inets.rwth-aachen.de

***Abstract*— In this paper we address the use of utility-based optimization in wireless networks. Our optimization approach is enabled by well-defined, technology- and platform-independent generic interfaces. The latter provide an abstract and unified representation of data and services available from the protocol stack, ranging from old legacy protocols to newer protocols. In particular, the application layer abstraction interface enables the expression of individual application requirements and reconfiguration of their tunable parameters. The proposed framework allows multiple applications, network protocols, and link layer technologies to coexist and evolve independently of each other. Therefore, it allows the resource manager to take more efficient decisions. Moreover, we present the analysis of a practical case study of utility-based optimization and the generated results. The source code for the application generic interface has been made available for public.**

I. Introduction

The rapid increase in the number of the wireless-enabled devices and their demands for high-speed multimedia communications makes the scarce spectrum resources very precious. Therefore, techniques related to Radio Resource Management (RRM) [1] that increase the efficiency of the use of radio resources are crucial for wireless communications. RRM techniques enable adapting the system to dynamic situations by controlling radio parameters such as data rates, channel allocation, and, thus, improve the overall system performance. The core functionalities of RRM include (a) reconfigurability that allows adjusting radio parameters at run-time, and (b) context-aware resource optimization with respect to, e.g., delay, throughput, spectrum efficiency, and QoS according to the end-to-end performance objectives.

In the last decade, cross-layer designs have been considered as important candidates for solving the problem of resource optimization since they allow information exchange across different layers of the protocol stack [2]. In such frameworks, the control entity that is responsible of resource management can build a comprehensive view of the status of the different layers. However, the interactions between these layers are complex and have to be performed only through standard interfaces. Otherwise, we need a structured and generic way to define such interactions and expose layered data in order to assist the optimization process. This way, cross-layer optimization modules could unleash the full use of information that resides at the various levels of protocol stack. Moreover, wireless networks can offer better experience to users if they can express their satisfaction and a way for the network to track it. One way to do this is to define utility functions that are specific for applications and users [3]. Further, these utility functions should be communicated to the resource manager in order to enable their use in the resource optimization process. In traditional networks, only standard utilities and policies can be used by the resource manager, or in the best case the operator can add new functions and policies. In order to make the implementation of utilities more flexible and allow applications to register their utilities at run-time, generic interfaces should be used between the resource manager and the applications.

In this context, we have been working on developing technologies which define standard generic interfaces [4] [5]. These interfaces offer technology-, platform- and protocol-independent way to access and control data and services available on the protocol stack. Furthermore, they can assist cross-layer and utility-based optimization techniques in the RRM. These generic interfaces are ULLA (Universal Link Layer API), GENI (Generic Network Interface) and CAPRI (Common Application Requirement Interface). ULLA abstracts the link/physical layer while GENI abstracts the transport/network layer. CAPRI is used to represent application requirements in terms of application layer utility that quantitatively expresses the end-to-end performance objectives.

In a related line of work there have been a lot of work on utility-based optimization and cross-layer optimization techniques in wireless networks. In this paper, we show how we can enhance the performance of the system by putting these two concepts together. More specifically, we show how we can exploit CAPRI functionalities in order to develop an optimization framework that allows the implementation of utility-based and cross-layer optimization for wireless networks. The major advantages of this approach are seamless data retrieval that can be used to evaluate various network configurations and the capability to differentiate between different application requirements. The former enables cross-layer optimization while the later facilitates utility-based optimization.

The remainder of this paper is organized as follows. In Section II we describe the motivation of this paper in more detail and provide some background. Then we explain the optimization framework in Section III, and provide an implementation case study in Section IV. We finally conclude the paper in Section V.

II. Motivation and Background

In traditional RRM approaches, each layer of the protocol stack conducts independent optimization processes. These processes obtain the required data using technology- or protocol-specific interfaces. Such approaches are obviously not flexible and cannot be used for implementing general and portable cross-layered RRM techniques where information from different layers are required.

To remedy the situation, various ways to abstract data and services in the protocol stack for easier access have been developed. Providing unified and generalized access to data and services across different layers and protocol entities is normally a key design requirement that enables such efficient optimization capability. Therefore, standard generic interfaces can be efficiently used to handle this issue by enabling seamless communication between protocol entities at different layers and radio technologies. In addition, these interfaces allow independent development of hardware and software. New functionalities of the RRM can be deployed on existing devices without the need to consult the Original Equipment Manufacturer (OEM).

In order to obtain the most efficient resource allocation solutions, the RRM needs to have a comprehensive representation of the environment and protocols' performance. In particular, the RRM needs to be aware of the different applications and their different goals [3]. These goals are normally quantitative and have to be translated to numerical expressions that can be understood by the resource manager. One possible technique is to use utility functions as a performance metric to model these optimization goals. Basically, utility functions combine several network attributes such as throughput or delay into numerical values depending on particular application requirements. This approach makes it possible to compare differences in network configurations and applications expressed by different utility functions. For instance, a downloading application is a throughput-sensitive application while a video streaming application is typically delay-sensitive. In such cases, applications can use the generic interfaces to express their optimization goals which can subsequently be used by the radio resource optimizer.

A. Generic Interfaces and Abstraction Layers

The concept of generic interfaces for computing platforms is well-known [6]. Generic interfaces usually describes a set of functions, parameters and objects, together with technology- and platform-independent application programming interfaces (APIs). The goal of these interfaces is to provide a unified representation of data and/or services available on the protocol entities. They act as an intermediate layer between the protocol entities and the control entities as depicted in Fig 1.

Generic interfaces are also used as a solution to the problem of how to incorporate newer protocol entities to older legacy protocols. Some vendors may use proprietary solutions by developing their own interfaces that are most optimized to their goals. However, these interfaces are typically not well-defined and only tied to specific vendors, and would not usually be fully interoperable with other products. By using a generic and unified interface, the system allows multiple applications, network protocols, and link technologies to coexist and evolve independently of each other.

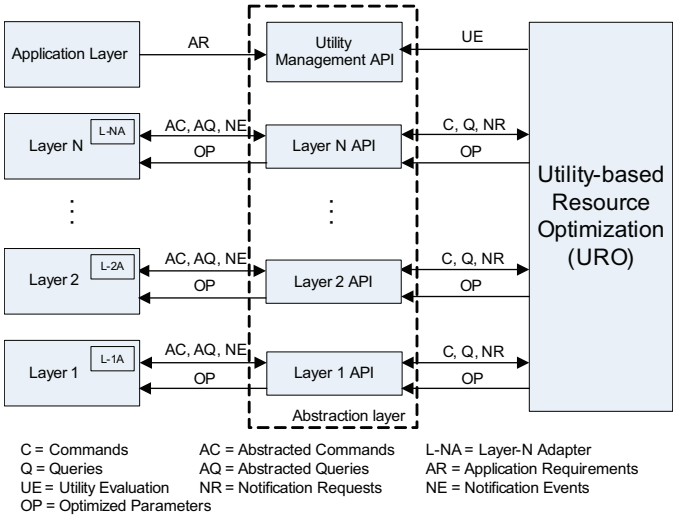

Fig. 1. Conceptual view of utility-based cross-layer optimization.

In this paper we extend and attempt to close the design gap of the concepts proposed in [7], [8]. Particularly, we do not only abstract data and services of the low layers of the protocol stack, but also express application requirements used to facilitate the resource management. Further, the framework design proposed in this paper is platform, operating system and protocol independent. We also take into account extensibility to support future applications, protocols, technologies and platforms, and as well as interoperability to support various types of devices with different resource capabilities including resource limited sensor devices, PDAs, and laptops.

The proposed framework bridges the OSI layers by providing protocol and application independent abstractions as shown in Fig. 1. The framework consists of two main parts, namely, abstraction layer and utility-based resource optimization.

The abstraction layer provides technology- and protocol-independent interfaces to synchronously retrieve data (Q), to support asynchronous notifications (NR) and to deliver control commands (C). The abstraction layer translates these requests into abstracted, technology- or protocol-dependent programming primitives (AC, AQ). Moreover, the powerful notification mechanism can either be event-based once a certain condition is fulfilled upon network changes, or periodic for updating layer information at specified intervals. The notification event (NE) is returned if one of the conditions is met. In case legacy protocols are deployed, Layer-N Adapter (L-NA) has to be used to cope with technology-specific ways to retrieve N-layer information and to provide transparency to the core of the abstraction layer. Moreover, via the abstraction layer, reconfiguration of layer parameters with the optimal values (OP) obtained from the resource optimizer is enabled. Finally, the utility management API, sitting on top of the abstraction

layer, provides capability of defining optimization goals in a quantitative manner. For instance, the applications register individual utility functions through this entity (AR).

The utility-based resource optimization (URO) is basically an extension of traditional RRM to support utility-based optimization. It is used to find the optimized parameters of each layer that maximize applications' utility value. The URO uses commands, queries and notifications exposed by the abstraction layer to retrieve the required information, either periodically or upon requests from the optimization process, to perform utility evaluation (UE) via utility management API.

B. Utility function and utility value

Fundamentally, the utility value is a quantitative representation of the quality of a connection measured at the application layer. It is used as a performance metric and optimization goal, and combines several network attributes such as throughput or delay depending on particular application requirements. A utility function can be defined as a function of one or more attribute, depending on application requirements. The definition of utility function becomes more complicated when multiple applications are simultaneously running and sharing the same communication link. In this case, we need an optimization mechanism that takes into account the system performance of these application interests.

A suitable solution to this problem is achieved through the utilization of utility functions that are evaluated based on the set of various measurable attributes of the connection. This gives a single real number, representing the utility of the connection for the application.

However, it is not trivial to combine these requirements into a single expression that can represent the total utility in order to perform the optimization process. In particular, we have to take into account that several applications can run on the same link and share resources, while each application can generate several data flows to/from various destinations/sources. Moreover, we have to consider that data flows may have different levels of priority that can be set by the administrator of the network or the user. In order to formulate these relations, we use the simple weighted sum approach based on objective weighting as presented in [9]. The weighted sum approach attempts to maximize the sum of the positively normalized, weighted, single objective scores of the parameter set solution. Each application is associated with a utility function that is evaluated based on the set of various measurable attributes of the connection. This gives a single real number, representing the utility of the connection for the application. Hence, we define a node utility $\bar{U}_n$ by

$$\bar{U}_n = \sum_{i=1}^{N} p^{(n,i)} U_{app}^{(n,i)}(\mathbf{a}_i) \tag{1}$$

where $U_{app}^{(n,i)}$ is the utility function of application app represented by data flow i and running on node n. This application has a nonnegative priority $p^{(n,i)}$ (high $p^{(n,i)}$ indicates high priority). In addition, $\mathbf{a}_i = (a_{i,1}, ..., a_{i,K})$ is a K-dimensional vector of network attributes characterizing the connection used by data flow i and N is the number of data flows running on the node. It should be noted that the summation of priorities is 1.

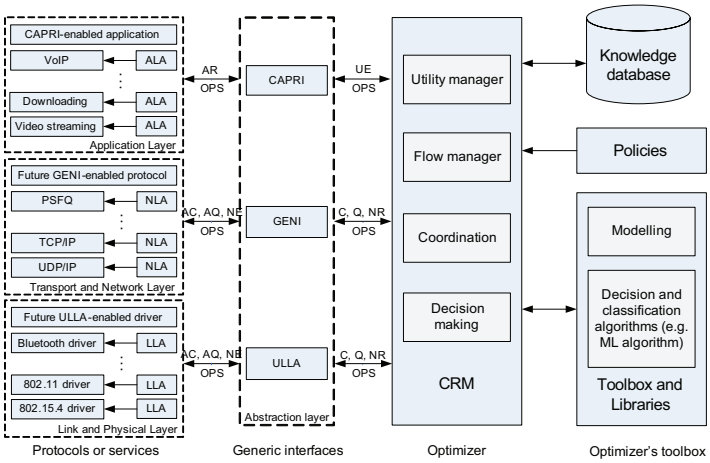

Fig. 2. Technical view of the proposed optimization framework.

$U_{app}^{(n,i)}$ is normally defined by the application provider. It can be a linear combination of separable functions of the different attributes [10]. It should be noted that our framework can handle any type of flow utilities but in our implementation we use a particular separable utility function defined by

$$U_{app}^{(n,i)}(\mathbf{a}_i) = \sum_{a \in \mathcal{A}} w_a^{(n,i)} u_a^{(n,i)}, \tag{2}$$

where $u_a^{(n,i)}$ is a utility function related to attribute a from set $\mathcal{A} = \{a_1, ..., a_K\}$, $w_a^{(n,i)}$ is a weighting coefficient of $u_a^{(n,i)}$ and $\sum w_a^{(n,i)} = 1$. The definition of $u_a^{(n,i)}$ and the value of $w_a^{(n,i)}$ are application-dependent and can be provided by the application developer or a third party. These utility functions are normalized to have values in the range [0,100] in order to have comparable order of magnitude of the utilities.

III. Framework Design

In this paper, we merge the concepts of utility functions and layer abstraction in Fig. 1 into a realistic and well-structured optimization framework as shown in Fig. 2. In our approach, we adopt the Cognitive Resource Manager (CRM) [11], a cognitive radio extension of traditional RRM approach, to perform as a core of the URO framework. In this framework, the CRM enables an implementation of learning mechanisms, complex control, and cross-layer optimization. These techniques are used to solve the resource management problems, in particular, in a multi-objective and multi-technology context. Furthermore, utility-based optimization is in fact a subset of complex control within the CRM toolbox. The implementation of the utility-based optimization framework for Microsoft Visual C++ is available from [5].

A. Components and interfaces

As shown in Fig. 2, the CRM is the core component of our optimization framework. The CRM has access to all the layers

of the protocol stack (for measurements and settings) via a set of well-defined APIs, and acts as a cognitive decision-making unit that performs local and/or global optimization based on information from the protocol stack, environmental readings or historical data utilized for learning and modeling.

The most important components of the CRM used in the optimization framework are (a) the *Flow manager* that provides information about available wireless adapters, established links and active flows and also defines interfaces for CRM to change configuration of adapters, links, and flows; and (b) the *Utility manager* that evaluates the utility value for each individual application and for the node, according to the network performance data; and (c) the *Decision-making component* that estimates the current network performance based on information from different layers, nodes, and historical data. This component then tries to identify the cause of problems and schedules actions to resolve them; and (d) *Policy engine* that receives defined policies from the policy server, reasons on them, and answers user's queries o spectral usage. The optional *knowledge database* can be utilized for improving learning and modeling mechanisms in the CRM toolbox. More details on the CRM components and implementation can be found in [12].

In order to optimize radio resources, the toolbox and libraries use information that are either stored in the knowledge database or gathered by the CRM core at run-time through the generic interfaces, namely ULLA, GENI, and CAPRI. The ULLA and GENI interfaces allow the CRM to interact with protocol stacks of different technologies. The CAPRI interface enables applications to express their requir ements and preferences in terms of network constraints and performance, which can be subsequently used by the CRM in its utility-based optimization process. Furthermore, by using information exposed by these generic interfaces, the proposed framework allows optimization across multiple layers including physical, data link, network, transport and application layers. Cross-layer optimization is performed within the CRM by using the exposed data from the different layers. Subsequently, a set of optimized parameters will be configured via ULLA, GENI, and CAPRI at the corresponding layers. It should be noted that ULLA, GENI and CAPRI are not dependent on the presence of CRM. They can be also used with any other resource manager if the data formats are respected.

To interface CRM with the protocol stack, for legacy protocols, abstraction layer adapters are required to translate programming primitives into protocol- or device-specific commands. In particular, the Link Layer Adapter (LLA) implements the ULLA link provider functions. The Network Layer Adapter (NLA) implements the GENI network provider functions. The Application Layer Adapter (ALA) abstracts away any application-specific details.

B. Specifying utility functions

CAPRI needs a concrete representation of utility functions. Utility functions can be specified in the form: "*register_utility*(flowID, utility_functions)" and can be specified by using mathematical expressions. The syntax consists of integers and real numbers, typical basic arithmetic operations ($+$ for addition, $-$ for subtraction/negation, $*$ for multiplication, $/$ for division, and $\hat{}$ for exponentiation), together with parentheses. Arithmetic functions such as natural logarithm (log()) and exponential function (exp()) are also included. For instance, the utility function $\log_{10}(t + 1000000)$ is a valid utility function for UDP data downloading where t is the measured throughput of this UDP flow.

In order to express simple conditions such as step function, we use the Iverson bracket notation. In this notation, the literal form [condition] is used, evaluating to 1 if the condition is satisfied and 0 otherwise. The conditions can be expressed using standard expressions such as $=$ (equal to), $!=$ (not equal to), $<$ (less than), $<=$ (less than or equal to), $>$ (greater than), $>=$ (greater than or equal to). Simple parentheses are also used if more sophisticated conditions need to be formed. Finally, measured values of different attributes of a flow, like throughput and delay, are used as the arguments of the utility function. In the prototype, we use t for throughput, d for delay, p for packet loss rate, and j for jitter.

For instance, if an application can only accept a maximum delay of 50 ms, the corresponding utility function can be presented as a step function, $[d <= 0.050]$.

The application can express this preference by combining the above specification with the previously mentioned utility function with different weights, as $0.7\log_{10}(t + 1000000) + 0.3[d <= 0.050]$,.

C. Utility-based optimization

The main objective of the resource management is to optimize the quality of the connections perceived by applications. For specified flow utility functions for flows running on a node, the optimization problem is to find a set of parameters $\mathbf{x} = \mathbf{x}_{opt}$ such that $\bar{U}_n$ is maximized. The optimal set of parameters can be formally defined as:

$$\mathbf{x}_{opt} = \arg\max_{\mathbf{x}\in X}\left\{\sum_i p_i U_{app}^{(n,i)}\big(\mathbf{a}_i(\mathbf{x})\big)\right\}, \tag{3}$$

where $\mathbf{x}$ is the cross-layer parameter tuple, X is the parameter space, and $\mathbf{x}_{opt}$ is the optimum parameter tuple that maximizes $\bar{U}_n$.

Each data flow i is associated with a utility function $U_{app}^{(n,i)}$ and a unique flow ID through the CAPRI interface. The utility is calculated based on network attributes retrieved periodically from the protocol stack through ULLA and GENI interfaces. The utility is used as a triggering mechanism for the optimization process. Alternative triggering mechanisms include changes in spectrum policy, flow priority, or link load. The utility-based triggering event reflects a change in the quality of connections as perceived by the CRM. We shall note that although the utility function is defined by the application, the utility is not used to maximize individual application needs, but rather to maximize the node or network performance as performed by the resource optimizer in the CRM.

Fig. 3 shows how utility-based optimization is realized in the URO framework. *CAPRI-aware applications* that explicitly use the CAPRI interface and link CAPRI library into them, can register their utility functions directly through CAPRI. On the other hands, the data flows of *legacy applications* that are not developed with the CAPRI interface and library, are detected by the flow manager once the applications generate data traffic. The CRM then automatically registers the newly detected data flow using predefined utility functions according to an application type, determined by the network ports the application is using. The registered utility functions of CAPRI-aware and legacy applications can be updated at run-time.

The *utility manager* calculates the utility values based on the information periodically collected by the CRM through the generic interfaces. The *decision engine* then examines the recent utility and triggers the configuration action if (a) $\bar{U}_n$ is below a certain threshold; or (b) the difference between $\bar{U}_n$ and moving average utility value is greater than a certain threshold; or (c) a periodic timer times out. It should be noted that the moving average threshold is used and periodically updated to prevent unnecessary activation of event triggering. If one of these conditions is met, the decision engine will notify the *action engine*. The action engine decides the configuration actions based on the current environmental conditions and resource available. These configuration actions involve network-layer traffic shaping, link-layer configuration and application-layer adaptation.

The *link-layer configuration* determines an appropriate action, such as channel switching and channel width resizing, which needs to be performed according to the current environmental conditions and application needs. For instance, if the application is a data-rate-hungry one, we may need to increase the bandwidth. However, if the used channel is relatively congested, it is better to move the transmission to some other channels with lower load. The *network-layer traffic shaping* determines the share of bandwidth for each application based on given application priorities. The *application-layer adaptation* determines which CAPRI-aware application parameters need to be tuned based on application priorities. As an example of configurable application parameters, we use data rate and encoding scheme. Link-layer configuration, network-layer traffic shaping and application-layer adaptation are performed through ULLA, GENI and CAPRI, respectively.

IV. Practical Case Study in IEEE 802.11 technologies

In this section we shall provide an implementation example to show how the proposed framework works in practice. The prototype has been implemented and ported on the IEEE 802.11-based WLANs. We validate the proposed framework deployed in a real-life scenario by porting two standard applications for it. The full description of implementation can be found in [13].

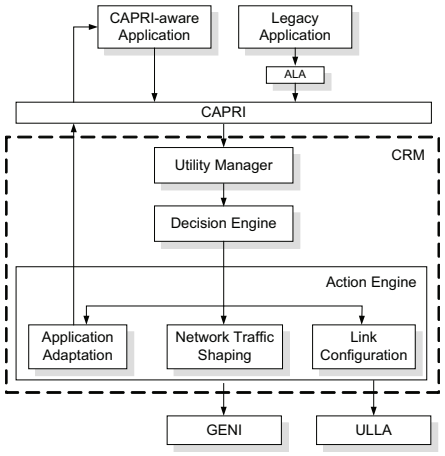

Fig. 3. Utility-based optimization in the CRM framework.

A. Scenario description

We have deployed a prototype setup of four standard notebooks to validate the proposed framework. The test scenario exploits two applications: UDP video streaming (VLC) [14] as a legacy application and UDP download application (DL) as a CAPRI-enabled application. Two users, called Alice and Bob in the following, are assumed to be within the same radio range.

The flow of the scenario is as follows. Alice wants to start downloading some photos from the photo library on her laptop while still watching the movie through local network. The video stream has a higher priority than the downloading stream. The priority level is set by the home network administrator, Alice in this case. Afterwards, Bob starts to synchronize his music folders with the music server. The spectrum occupancy is rather high due to the fact that Alice and Bob's house is surrounded by a lot of wireless networks and therefore there is only one channel (i.e. the used channel) available. The main objective is to maximize the video quality perceived by Alice while Bob's download stream is running and sharing the medium.

When all these applications are running, the CRM discovers that the current link is overloaded. However, after checking some performance indicators through ULLA and GENI, it does not detect that there is external interference. The CRM also knows that the link is already using the maximal channel width, and thus there is no space to expand. Therefore, it will re-prioritize the resource assignment to ensure that the high priority application, VLC, receives sufficient bandwidth.

B. Optimization performance

We define the utility function of the VLC application as a function of throughput ($u_t^{(n,i)}$) and delay ($u_d^{(n,i)}$). Video streaming application is considered as a delay-sensitive application. Therefore, $w_d^{(n,i)}$ is set to a higher value of 0.8 while $w_t^{(n,i)}$ is set to 0.2, and thus $U_{\text{VLC}}^{(n,i)} = 0.2 \cdot u_t^{(n,i)} + 0.8 \cdot u_d^{(n,i)}$.

TABLE I

OPTIMIZATION PERFORMANCE.

	T_1	T_2	T_3	Change from T_2 to T_3 (%)
Interference	low	high	high	
Optimization	disabled	disabled	enabled	
$d_{\text{AVG}}^{(\text{Alice})}$	7.8 ms	36.5 ms	14.5 ms	↓60%
$t_{\text{DL, AVG}}^{(\text{Alice})}$	9 Mbps	6 Mbps	1 Mbps	↓600%
$t_{\text{VLC, AVG}}^{(\text{Alice})}$	6 Mbps	3 Mbps	5.5 Mbps	↑250%
$t_{\text{DL, AVG}}^{(\text{Bob})}$	-	8 Mbps	9 Mbps	↑12.5%
$\bar{U}_{\text{Alice}}$	94	60	95	↑58

We define $u_t^{(n,i)}$ by

$$u_t^{(n,i)} = 100 \cdot \frac{\log_{10}(\min(t, R_{\text{max}}) + 1000000) - 6}{\log_{10}(R_{\text{max}} + 1000000) - 6}, \quad (4)$$

and $u_d^{(n,i)}$ by

$$u_d^{(n,i)} = \begin{cases} 100 - 0.4d, & \text{if } d < 5, \\ 75 - 25\tanh(\frac{d-12.95}{5}), & \text{if } 5 \leq d < 25, \\ 57 - (0.264d), & \text{if } 25 \leq d < 215 \end{cases} \quad (5)$$

where t is a throughput measured in bits/s, d is delay measured in milliseconds and maximum allowed data rate R_{max} can be defined by the application when registering to CAPRI or adjusted during run-time by the CRM through CAPRI. In our settings, we consider that R_{max} = 9 Mbps at the beginning.

The DL application is considered as a function of throughput $U_{\text{DL}}^{(n,i)} = u_t^{(n,i)}$ only. From (1), we compute the node utilities perceived by Alice and Bob as $\bar{U}_{\text{Alice}} = \frac{2}{3}U_{\text{VLC}}^{(\text{Alice},1)} + \frac{1}{3}U_{\text{DL}}^{(\text{Alice},2)}$ and $\bar{U}_{\text{Bob}} = U_{\text{DL}}^{(\text{Bob},1)}$.

The utility functions used in this example are modified versions of functions in [15] so that the weighting coefficients and thresholds are suitable for our prototype. In this scenario, we consider that VLC has a higher priority than DL. First, Alice starts VLC and DL. Thereafter, at time T_1 Bob starts his download stream. The optimization is triggered at time T_2. Table I shows that when Bob starts his application the average delay d_{AVG} is almost four times higher than before, causing some glitches on Alice's video and degradation of Alice's node utility. However, when the optimization process is applied, the maximum allowed data rate of Alice's DL is lowered from 6 Mbps to 1 Mbps, resulting in lower delay and higher data rate in Alice's VLC and Bob's DL. Additionally, Alice's node utility increases by 58%. Alice is unaware of this action, although she notices that her downloading stream is somewhat slower. Alice is now satisfied as the movie is running smoothly again after a short period of degradation.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a utility-based optimization framework for wireless networks that enables cross-layer optimization at run-time through well-defined generic interfaces. In particular, the proposed framework allows (a) the resource optimizer to seamlessly retrieve data and services via ULLA and GENI interfaces; (b) the applications to express their requirements by specifying different utility functions via CAPRI interface; and (c) the utility-based resource optimization to use information from all OSI layers to maximize the network utilization across multiple layers. The optimized parameters are reconfigured in the protocol stack via ULLA, GENI and CAPRI interfaces.

We have validated the prototype implementation of the proposed framework in a real-life scenario on the Windows platform. The results have shown that the proposed framework maximizes the network utilization in a non-cooperative network by reconfiguring the parameters according to the decision made by the optimization process. The implementation of the proposed framework for Microsoft Visual C++ is available from [5].

ACKNOWLEDGMENT

The authors thank the RWTH Aachen University and the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) for providing financial support through the UMIC research center. We also thank the European Union for providing partial funding through the FARAMIR project.

REFERENCES

[1] J. Zander, "Radio resource management in future wireless networks: Requirements and limitations," *Communications Magazine, IEEE*, vol. 35, no. 8, pp. 30 –36, aug 1997.

[2] E. Hossain, D. Niyato, and Z. Han, *Dynamic spectrum access and management in cognitive radio networks*. Cambridge U. Press, 2009.

[3] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176 –1188, Sep. 1995.

[4] *GOLLUM project*, http://www.ist-gollum.org, last visited 30/04/2011.

[5] *ARAGORN project*, http://www.ict-aragorn.eu, last visited 30/04/2011.

[6] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Comput. Commun. Rev.*, vol. 29, pp. 7–23, April 1999.

[7] J. Riihijärvi, M. Wellens, and P. Mähönen, "Link-layer abstractions for utility-based optimization in cognitive wireless networks," in *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, 2006.

[8] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer, "Application-driven cross-layer optimization for video streaming over wireless networks," in *IEEE Communications Magazine*, vol. 44, no. 1, Jan. 2006.

[9] C. Fonseca and P. Fleming, "Multi-objective optimization and multiple constraint handling with evolutionary algorithms - I: A unified formulation," *IEEE Transactions on Systems, Man and Cybernetics*, 1998.

[10] C. L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making Methods and Applications*. Springer-Verlag, 1979.

[11] P. Mähönen, M. Petrova, J. Riihijärvi, and M. Wellens, "Cognitive wireless networks: your network just became a teenager," in *Proceedings of the 25th Conference on Computer Communications*, 2006.

[12] E. Meshkova *et al.*, "Using cognitive radio principles for wireless resource management in home networking," in *IEEE Consumer Communications and Networking Conference (CCNC 2011)*, USA, 2011.

[13] K. Rerkrai, J. Nasreddine, Z. Wang, J. Riihijärvi, and P. Mähönen, "Design and Implementation of Utility-Based Radio Resource Optimization Using CAPRI," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2011.

[14] *Video LAN project*, http://www.videolan.org/vlc, last visited 30/04/2011.

[15] C. Boutremans and J.-Y. Le Boudec, "Adaptive delay aware error control for internet telephony," in *Proc. of 2nd IP-Telephony Workshop*, 2001.