EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

Context-aware resource allocation for cellular wireless networks

Magnus Proebster[1*], Matthias Kaschub[1], Thomas Werthmann[1] and Stefan Valentin[2]

Abstract

Current cellular networks are often overloaded by Smartphone traffic, while the users' Quality of Service (QoS) demands are not met. To cope with this problem, we demonstrate a new radio resource management approach. With *Context-Aware Resource Allocation*, the base station's scheduler (i) observes *Context Information (CI)* from the user's environment and (ii) utilizes this knowledge for an efficient throughput-delay tradeoff. After introducing our framework for accessing CI from the handheld's applications and operating system, we use time-utility functions to develop a practical scheduling algorithm. Studying this heuristic under various traffic assumptions shows that our context-aware scheduler can support three times the load of proportional fair scheduling, at equal capacity and utility. Thus, even a small degree of CI increases the wireless links' efficiency without sacrificing the users' QoS.

Keywords: Radio resource management, Scheduling, Cross-layer design, Cellular wireless networks, Context-aware, Quality of service

Introduction

Modern User Equipments (UEs), like *Smartphones* and *Tablets*, lead to serious load problems in radio networks [1]. This has three reasons. First, users of modern UEs generate higher average load [2] and spend longer time with their devices than with traditional cellphones [3]. Second, modern UEs lead to traffic patterns that are difficult to model and to predict. Such ambiguous traffic statistics do not only result from the bursty Internet traffic [4] but also from multitasking operating systems, which allow the users to quickly change the active applications. Third, as most of these applications are using a single IP layer, the base station's Radio Resource Management (RRM) is not aware of the QoS requirements of the running applications. This is the starting point for our approach, called Context-Aware Resource Allocation (CARA).

Context-Aware Resource Allocation copes with the intense load of modern UEs by making the RRM aware of application demands. Currently, the RRM cannot distinguish between the traffic of urgent and delay-tolerant applications. However, in current radio networks, a significant portion of the traffic can wait [2]. For instance, downloads of firmware upgrades or traffic of other background applications can be suspended in case of congestion. If the base station's RRM would be aware of the delay requirements of each application, large QoS gains can be achieved or higher network load can be supported.

Idea

Context-Aware Resource Allocation is based on three key components to (i) obtain Context Information (CI) at the UE, (ii) signal this information to the base station's RRM, and (iii) to efficiently use the CI for RRM. This third component is the focus of our article.

In particular, we design a wireless scheduler that exploits information on the application's QoS requirements and from the UE's operating system. For each application, this scheduler is aware of the corresponding packet flow and the delay requirements derived from CI, e.g., the state of the application window (i.e., either in the background or the foreground of the screen). The scheduler then uses this information to prioritize the applications' flows according to their window state and their delay budget. As a result, the scheduler performs a throughput-delay tradeoff.

*Correspondence: magnus.proebster@ikr.uni-stuttgart.de
[1] Institute of Communication Networks and Computer Engineering, Universität Stuttgart, Stuttgart, Germany
Full list of author information is available at the end of the article

Contributions

Following this idea, our contributions are:

- *Transactions:* This logical unit contains all data transmissions leading to a user observable result at the application layer. With transactions, a scheduler can assign resources to complete application flows instead of only for the upcoming Transmission Time Interval (TTI).
- *Deadline-based weighting:* We derive scheduling weights from time-utility functions [5] based on an application's local deadline. For instance, a change of the application window state may lead to a different deadline and, thus, a different time-utility function. This soft-QoS approach operates on transactions instead of individual UEs. Thus, it can weight multiple applications individually even if they run in parallel on a single UE.
- *Scheduling algorithm:* Based on the above utility functions, we formulate the scheduling problem as a Linear Program (LP) and design a scheduling heuristic using ideas from evolutionary algorithms.
- *Extensive study using accurate models:* Our results clearly demonstrate the fast convergence and high performance of our scheduling heuristic for various traffic situations. We obtain these results under realistic traffic, channel, and interference assumptions on the base of upcoming 3GPP Long Term Evolution (LTE) systems.

All in all, this work provides a practical RRM scheme that exploits new information from the user's environment for more efficient scheduling. Rigorous studies show that the approach is efficient, practical and easy to extend to further CI.

Related work

The related work falls in the fields of *QoS differentiation* and *utility-based scheduling*. For QoS differentiation, various approaches exist on the IP layer such as IntServ, DiffServ or via RSVP [6]. In cellular networks, the Medium Access Control (MAC) employs different Access Point Name (APN) bearers to distinguish between packets of different service classes. Nonetheless, none of these approaches represents individual delay constraints of applications running in the same traffic class.

Defining scheduling weights via utility functions has become a standard approach in RRM [7]. Typically, strictly concave utility functions are employed to account for fairness constraints within the objective function [8]. Convex formulations of the subcarrier and power allocation problem have lead to efficient heuristics even for multiple traffic classes [9,10]. However, most utility-based schedulers operate on average data rates and can, thus, not directly account for delay constraints.

Such constraints are included by the time-utility functions in [5], which is closest to our work. Here, Ryu et al. use delay-dependent utility functions to derive a priority factor. This factor is then included into classic Proportional Fair (PF) scheduling. Unlike our work, this method (i) can only consider the current situation in a TTI and has no knowledge of the packet flows, (ii) is based on traffic classes and, thus, cannot account for the individual demands per application, (iii) expresses utility as a truncated exponential function, whose discontinuity can lead to infeasible solutions. Moreover, Ryu et al. [5] clearly state, their algorithm leads to substantially lower throughput than the PF scheduler. As shown in Section "Performance evaluation", this is not the case with our context-aware scheduler, which even increases the channel's degrees of freedom by planning resource allocation multiple time slots in advance.

Structure

In Section "Context-Aware Resource Allocation", we describe the framework for CARA by defining CI, transactions and utility functions. Then, we formulate CARA as a LP and discuss its feasibility in Section "Scheduling problem". Section "Practical scheduling heuristic" presents our scheduling heuristic which can be implemented in a real base station. In Section "System model", we describe the system model for our simulation studies. Simulation results are then presented in Section "Performance evaluation", which provides throughput and QoS gains compared to common scheduling schemes. After studying the complexity and convergence of our algorithm, we conclude the article in Section "Conclusion".

Context-Aware Resource Allocation

Context Information

We use the term CI to describe the knowledge a base station can get about data transmissions, which is relevant for scheduling. With CI, we want to know how delay and accordingly the scheduling decisions influence the QoS and the user's perception of an application. As sources for CI, we distinguish different *context features*, for example about the users' environment or about the urgency of a transmission. For instance, the following context features are relevant for scheduling:

- Application window state: whether an application is currently displayed in the foreground
- Type of application: for instance interactive applications or system applications without user interface
- Type of request: whether the requested data is for caching or it is to be displayed immediately

The user's perception is closely related to application layer measures which aggregate effects from the lower layers. Therefore, important CI exists in the application running on the UE. The application knows the event triggering a certain action; the application will likely know which type of data it is transmitting or requesting; and the application often knows if the user is waiting for feedback.

Moreover, the operating system and the platform libraries are aware of the screen saver, the UE orientation, timers, touchscreen events, battery state, and other device parameters which help to determine the latency requirements of the network transmissions. Furthermore, the user could configure preferred applications on the UE which could then be prioritized for scheduling.

Our definition of CI also contains Channel Quality Information (CQI) which is already exploited in most state-of-the-art schedulers.

The sources of CI mentioned so far are only available in the UE. Additionally, the base station or the radio access network know about the current network load. Furthermore, the radio access network can apply Deep Packet Inspection (DPI) to estimate file-types, data size, and similar information available in the protocol headers. However, DPI is limited for encrypted packets and the base station cannot extract context features such as the application window state at all.

By using CI, CARA differs from most other approaches that use measurements on network level, such as bandwidth, packet-delay, and jitter. We extend classical cross-layer approaches by directly applying CI from the application layer and concentrating on effects visible to the user. This allows to also distinguish between different applications running on a single UE and using the same protocol.

For our approach, we focus on context features that can be derived from knowledge in the applications, the operating system, and the radio interface of the UEs. In particular, we exploit the following context features:

- The QoS demand of the application with respect to delay. This is expressed by the shape of the time-utility function.
- The point in time when the user is expecting a transaction to be finished, which we call *expected finish time*. This will parametrize the utility function.
- A prediction of the amount of data that will be transmitted for a transaction.
- The CQI, which is already in use in other opportunistic radio network schedulers.

In today's Smartphone eco-systems, such context features can be obtained easily. There is a small number of relevant platforms (e.g., iOS, Android) which could provide an Application Programming Interface (API) for collecting CI. Even without implementation in additional applications, many cases of network transmissions could already be covered by enhancing the built-in services like web-browsing engine and download manager, for example. Of course, in a CARA-enhanced network, there is also an incentive for application developers to signal knowledge about the application's network transmissions, where possible, to improve the user experience. We discuss the signaling of CI in Section "Signaling CI"

Transaction framework

To improve scheduling in terms of QoS, the extracted CI has to be linked to the data that is actually transmitted. This is done by the following transaction framework.

Figure 1 shows the architecture of the transaction framework. Data transmissions are assigned to transactions which contain all data from a request to a user observable result. The applications, platform libraries, operating system, and UE collect CI. A transaction manager collects and aggregates this CI and derives transaction requirements which are signaled to the base station. The base station separates traffic into transactions and uses CI for its scheduling decision.

For the definition of transactions, we look at today's Internet traffic. Large-scale measurements in a commercial access network show that 57% of Internet traffic uses Hyper-Text Transfer Protocol (HTTP) [11]. Most of the remaining traffic behaves similarly, notably e-mail protocols, Network News Transfer Protocol (NNTP) and various Peer-to-Peer (P2P) protocols. To account for this type of web traffic, we assume that the amount of data to transmit is fixed for one object. When this amount of data is transmitted, the object is finished. Secondly, we assume that the transmission is elastic, i.e., it adapts to the available bandwidth. Since the amount of data is fixed, the finish time depends on the available bandwidth.

For such *web-like* traffic, we define a transaction to be the complete set of data that needs to be transmitted for a user-observable result. For example, downloading a webpage with all embedded elements is a transaction, the result is displaying the web page. Thus, a transaction can consist of multiple Transmission Control Protocol (TCP) flows or be only a part of a single flow.

Unlike common network metrics such as bandwidth and packet delay, transactions can directly express the user's interactions and perceptions for web-like traffic. However, media streaming is not covered by this definition and has to be handled differently. In this article, we do not consider streaming traffic, but focus on web-like traffic using HTTP and FTP. Furthermore, we disregard applications like voice calls here, as their strict delay requirements leave practically no room for shifting them in time. Consequently, real-time traffic would essentially reduce the cell capacity available for CARA scheduling.

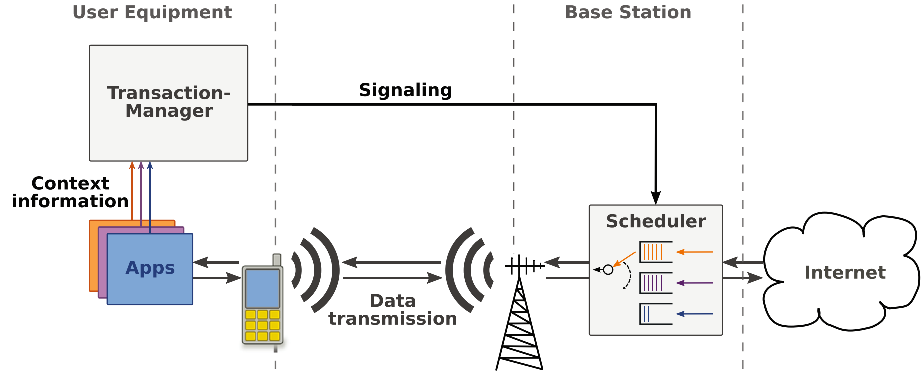

Figure 1 **Architecture of the context-aware scheduling approach.**

Fairness among users can also be provided with this approach. The base station collects CI from all users and assigns the resources based on this information. That means that, finally, the scheduling relies on information provided by the user. This could lead to users trying to cheat to get a better data rate. However, the base station could detect anomalies in user-reported CI and even reduce the priority of users reporting exaggerated requirements. By this, it can be ensured that users reporting always urgent transmissions are not treated preferentially.

The scheduler, usually operating on MAC frames, needs a mapping between frames and transactions to jointly treat a whole transaction. For example, in LTE networks traffic arrives at the scheduler in the form of Internet Protocol (IP) packets. Consequently, we need a classification of IP packets to transactions. This can be mostly done by header inspection using the TCP/IP five-tuple. We can achieve an even finer granularity by adding starting and ending bytes of the TCP stream.

Where the classification is done depends on the signaling scheme in use, as described in the following paragraph. If the UE directly signals to the base station, the base station also has to perform the packet classification. However, if this is infeasible, e.g. due to limited processing capabilities, CI can be signaled to the packet gateway, which is foreseen in LTE to be able to perform DPI. These DPI capabilities are sufficient to classify IP packets into transactions.

Signaling CI

Signaling CI from the UE to the base station is an important prerequisite for CARA. In this section, we describe signaling variants for LTE systems. The signaled information is composed of the identification of transactions, e.g. the TCP/IP five-tuple, and the transactions' requirements.

A straightforward approach is to signal CI directly between UE and eNB over a dedicated control channel above the Radio Link Control (RLC)-layer. As the eNB always is the next hop from the UE, no routing and addressing is required. An alternative would be signaling in the data plane and addressing the eNB via IP anycast. In this case, no additional control channel needs to be standardized and existing UEs could be updated in software. However, both alternatives fall short, if data packets are encrypted between packet gateway and UE and cannot be classified in the eNB, or if the eNB's processing capabilities are too limited.

Therefore, another variant includes the packet gateway in the signaling path. With this approach, the UE just sends signaling packets to its next IP destination, the packet gateway. With the signaled identification information, the packet gateway classifies incoming packets and sets a different DiffServ Code Point (DSCP) for each transaction. It uses a reserved DSCP to forward mapping information to the eNB. The LTE standard does not define a signaling path from the packet gateway to the eNB. However, the usage of DSCPs is straightforward, as the LTE backhaul has a flat IP-architecture and DSCPs are already employed for the classification of radio bearers. The packet gateway furthermore forwards the information about the transactions' requirements to the eNB with the reserved DSCP.

Each signaled transaction gets an identifier. If changes in the requirements of a transaction occur, e.g. because the respective application was sent to the background or more precise information about the remaining size is available, the UE can signal updated CI.

We approximate the size of signaling messages in an example implementation, using a simple text-based protocol, to be in the order of 50 Bytes. When assuming an average transaction size in the downlink of 30 kb, the uplink signaling overhead compared to the downlink is 0.2%. Even with several information updates per transaction, the overhead remains low.

From application layer information to utility functions

To apply CI for scheduling, we give each transaction a utility function. We use these utility functions to determine relative weights for all active transactions and then schedule the transaction with the largest weight.

The utility functions express the latency requirements of a transaction, i.e., the QoS degradation over waiting time. We derive the shape and the parameters of the utility functions from CI. For instance, using the application window state, we can say that the QoS of a transaction belonging to a foreground application degrades earlier than the QoS of transactions for background applications. When surfing the web, users appreciate fast page loads, but also tolerate a certain delay [12,13]. Such soft QoS degradation with respect to delay is expressed in our utility functions.

Knowing the latency requirements of transactions from utility functions allows the scheduler to decide which transaction should be scheduled when. In many approaches from literature, utility is defined as a function of the data rate [7-9]. However, here we want to express the value of a transaction for the user. For web-like transactions, this depends on the finish time only. We define a transaction's utility U to be in the interval [0, 1], where 1 is the optimal utility achieved for zero delay and 0 is the worst utility which is reached after infinite delay. The interval is closed because we assume that the level of the user's satisfaction or dissatisfaction is limited. That is, the service cannot be better than "excellent" or worse than "annoying". The user accepts a certain waiting time for the completion of a request. If a transaction is finished earlier than expected, this can only slightly increase its value. This corresponds to [13], which found that delays of less than 0.1 s are not noticeable in interactive applications, e.g., showing a web page. If the transmission takes longer than expected, the transaction's utility decreases. We assume that all users eventually abandon their transactions if they have to wait significantly longer than the expected finish time [12]. For these users, the level of annoyance cannot further increase. Hence, for an increasing finish time the utility of the transaction converges to 0.

The resulting function of the transaction's utility has an inversed S-shape. To provide this shape, we choose the logistic function

$$U(t) = \frac{1}{1 + e^{(t - t_{\text{infl}})k}} \tag{1}$$

where t is the finish time of the transaction, t_{infl} is the inflection point, and k scales the steepness of the curve. Note that other inversely S-shaped functions would also be possible to model the user's QoS. Figure 2 shows an exemplary utility function with its parameters. Here, we give a general definition of the requirements for a range of applications. We will model different types of applications by different parameterizations. Our approach also allows to define different utility functions and to directly set appropriate latency requirements.

We obtain the parameters in (1) by

$$t_{\text{infl}} = t_{\text{start}} + x \cdot (t_{\text{exp}} - t_{\text{start}}) \tag{2}$$

$$k = \frac{d_{\text{exp}}}{(1-x)L} \ln\left(\frac{1}{U(t_{\text{exp}})} - 1\right) \tag{3}$$

which will be explained in the following.

We assume that the transaction arrives at the scheduler at time t_{start}. All other points in time are relative to t_{start}. We define the utility of a transaction finished in the time expected by the user to be $U(t_{\text{exp}})$. Because the utility slightly increases, when the transaction finishes earlier than expected, $U(t_{\text{exp}})$ is less than 1.

The expected finish time of a transaction depends on its size, on the type of application, and on the user's context. Here, we assume that the user expects a certain data rate d_{exp} from the operator. This data rate depends on the type of application currently in use. The user expects the service to run satisfactory which means that foreground applications usually require a higher data rate than background applications. The expected finish time of a transaction is then determined by

$$t_{\text{exp}} = t_{\text{start}} + \frac{L}{d_{\text{exp}}} \tag{4}$$

where L is the length of the transaction in bits and d_{exp} is the expected data rate in bits per second.

To adjust the steepness of our inversely S-shaped utility function, we vary the relation between t_{exp} and the inflection point of the logistic function, $U(t_{\text{infl}}) = 1/2$, with an arbitrary scaling factor x. Equation (3) is obtained by inserting $U(t_{\text{exp}})$ into (1) and solving for k. With these parameters, the utility function is completely defined and provides a continuous mapping of utility in terms of transaction finish times. The resulting $U(t)$ is only counted for finished transactions. Dropped or aborted transactions always result in $U = 0$. Example values of the selectable variables $U(t_{\text{exp}})$, d_{exp}, and x will be given in Section "System model".

Deployment issues

Operators can deploy CARA step by step. Initially, CARA works without UE assistance. By using context features available at the network side and DPI, which is already used in LTE networks, the scheduler can identify transactions and assume default requirements and transaction sizes based on the content type. Although such a solution already outperforms state-of-the-art schedulers by meeting finish times, it does not exploit the full

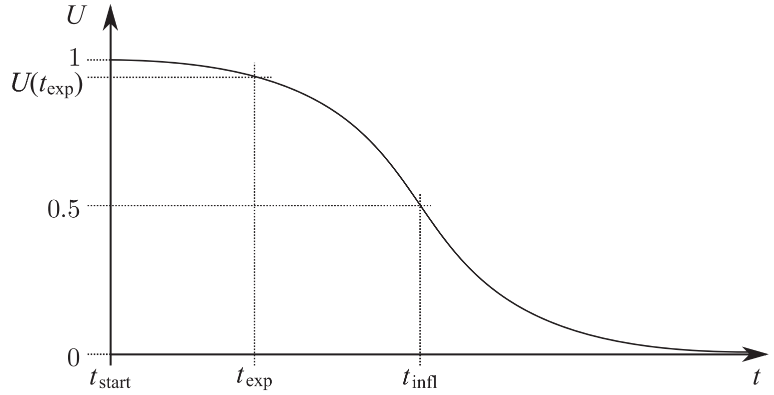

Figure 2 Parameters of the utility functions.

potential of CARA. Most context features, e.g. the foreground/background state of an application, are only available in the UE and cannot be exploited by the scheduler without signaling.

When introducing signaling, already the first application or UE can profit from CARA's signaling. Now the scheduler is aware of the exact requirements of the respective transactions and aims to fulfill them. This provides a QoS gain even if only one UE supports CARA.

Practically speaking, CARA can be deployed by a simple programming library or handheld Operating System (OS) extension. Soon, as the advantages become clear, more and more application developers may use a context signaling library, which gives them an advantage over competitors.

There is also no problem, when a CARA enabled UE enters a cell without CARA support. Signaling messages will just be dropped in the network and conventional scheduling is used.

In Section "Performance with deteriorated CI", we evaluate the performance of CARA with deteriorated or missing CI to investigate scenarios without full CARA support.

Scheduling problem

Context-aware resource allocation aims to improve resource allocation by CI. To perform such resource allocation, we need to decide in each TTI which resources shall be given to which user. Basically, we need an instantaneous weighting for transactions to perform the actual scheduling decision. The difficulty is that these instantaneous decisions have to lead to the desired behavior in the long run. This requires a stateful scheduling algorithm with a consistent behavior over several TTIs.

A straightforward algorithm for CARA would be a transaction-based Earliest Deadline First scheduler. This means that we schedule the transaction with the closest deadline. As deadline, we could choose a certain utility threshold. However, such a simple scheme cannot exploit the full potential of CARA. For example, it ignores channel conditions and object sizes.

Let us now formulate CARA as a utility maximization problem and point out the difficulties to solve the problem by an LP.

Problem formulation

The task of CARA is to maximize the utility of the network for the given resources. We can formulate CARA as a utility maximization problem to get the optimal performance for a given set of transactions and known channel quality [14]. The objective is to maximize

$$U_{\text{total}} = \sum_{n} \sum_{t} \delta_n(t) U_n(t) \tag{5}$$

where U_n is the utility function of transaction n. This function depends on the transaction finish time $t_{\text{fin, n}}$ which results from the chosen resource allocation. The binary vector $\delta_n(t) \in \{0, 1\}$ is a "flag" which is one for the finish time $t = t_{\text{fin, n}}$ of transaction n and zero for all other t. The transaction index is $n \in \{1, \ldots, N\}$ for N transactions in the system.

The decision variables of the above objective function represent the resource allocation. The resources $r_n(t)$ are allocated to transaction n in TTI t and lead to a certain order of the finish flags $\delta_n(t)$. The optimization problem (5) has the following constraints:

$$\forall n: \sum_{t} \delta_n(t) = 1 \tag{6}$$

$$\forall n, t: \delta_n(t) \leq \frac{1}{L_n} \left(\sum_{t_1=1}^{t} r_n(t_1) c_n(t_1) \right) \tag{7}$$

$$\forall t: R \geq \sum_{n} r_n(t) \tag{8}$$

$$\forall n, t < t_{\text{start, n}}: r_n(t) = 0 \tag{9}$$

Here, $c_n(t)$ is the channel quality of transaction n at time t in bits/resource. L_n is the length of transaction n in bits. The time slots are $t \in \{1, \ldots, N_{\text{TTI}}\}$. Parameter $t_{\text{start, n}}$ is the start of transaction n and R is the number of resource units.

Constraint (6) enforces that there is exactly one finish flag for each transaction. The finish flag is defined by (7) stating that a transaction can only be finished after all of its data has been transmitted. To do so, the data transmitted in each TTI is calculated by the product of channel quality and allocated resources. With (8), we constrain the resource allocation to the available resources. Finally, (9) ensures causality, i.e., no transaction gets resources before t_{start}.

Throughout the rest of the article we remove the frequency dimension from the problem formulation by setting $R = 1$. This improves clarity, as the effects of frequency selectivity are well-understood [15] and are not necessary for a first investigation of CARA.

Solution with LP

The above optimization problem (5) to (9) can be solved as an LP. Unfortunately, solving this LP is computationally expensive. The decision variables contain the resource allocation of N active transactions for each TTI. The number of decision variables is the cardinality of the sets $r_n(t)$ and $\delta_n(t)$, which is $|\{\forall n, t : r_n(t)\}| + |\{\forall n, t : \delta_n(t)\}|$. With N_{TTI} TTIs, this leads to $2 \cdot N \cdot N_{\text{TTI}}$ variables. The integer constraint expressing the finish time of transactions (7) is the linearization of a maximum operator of the binary condition, which states if a transaction is finished or not. This renders the whole optimization problem ill-conditioned such that an LP would have to investigate the whole solution space. As a consequence, this option is infeasible for a reasonable number of transactions on typical hardware.

Furthermore, the solution space and the relationship between neighboring solutions is very scattered. A slight change, like the exchange of two transactions, for example, may lead to a completely different U_{total}. Because of that, also evolutionary algorithms have a slow convergence behavior and cannot find an optimal solution reliably [14].

Due to these problems of the optimal solution, we developed a practical scheduling heuristic which solves (5) efficiently. This heuristic will be described in the following section.

Practical scheduling heuristic

Our scheduling heuristic aims to increase the sum utility of all transactions with less computational complexity than linear programming. The heuristic consists of two steps. The first step determines the sequence of the scheduled transactions and the second step combines this sequence with PF.

The intention behind our algorithm is that alternating between multiple active transactions leads to longer finish times and, thus, reduces the total utility [14]. In our approach, we improve the finish times by reducing this fragmentation. In the first step, we create a scheduling sequence of all active transactions. For this sequence, it is assumed that a transaction is only served after the previous one is completed. An evolutionary algorithm searches for the sequence that delivers the maximum sum utility. It starts with an arbitrary sequence and mutates this sequence for a number of iterations in order to find a sequence with a higher sum utility. Our restriction to only serve complete transactions sustainably reduces the solution space of the algorithm improving the sequence.

To avoid drawbacks from this simplification, we combine the sequence with an ordinary PF scheduling weight in the second step. This may change the scheduling order, if it is advantageous to schedule a different transaction in the current TTI. We combine the sequence and the PF scheduling weight by giving transactions a penalty that increases with their scheduling order and subtract this penalty from the original PF scheduling weight. If we would always stick to the strict sequence order, this would reduce the scheduling granularity to the granularity of transactions. Then, large transactions would reduce the possibility to react on fast channel variations. By combining the transaction order with PF weights, we allow to prefer a different transaction in the short term if this is indicated by the channel conditions.

Finally, the transaction with the highest weight is scheduled once per TTI. In the following, we formalize the two steps.

Step 1: sequence determination

In step 1, we determine a sequence of transactions by improving the total utility iteratively.

(1) Start with an arbitrary sequence of transactions $S_1 = \{n_{11}, n_{12}, \ldots, n_{1N}\}$ with n_{ij} being the transaction at index j in sequence i.
(2) Determine the total sum utility $U_{\text{total}}(S_1)$ of S_1 as described in Algorithm 1.
(3) Mutate sequence S_1 into sequence S_2 with the shift-operation given in Algorithm 2.
(4) Calculate $U_{\text{total}}(S_2)$ with Algorithm 1.
(5) Keep the sequence with better utility and repeat from (3) for a predefined number of iterations N_I as described in Algorithm 3.

Algorithm 1 Utility determination

```
$j = 1$
$l_j$ = remaining bits of transaction $j$
$t$ = current time
$U_{\text{total}} = 0$
while $j <= N$ do {for all transactions}
  $l_j = l_j - c_j(t)$
  if $l_j <= 0$ then {no more bits to transmit}
    $U_{\text{total}} = U_{\text{total}} + U_j(t)$
    $j = j + 1$
  end if
  $t = t + T_{\text{TTI}}$
end while
return $U_{\text{total}}$
```

Algorithm 2 Mutation by shift operation

```
Choose any two indices $u$ and $v \in \{1, \cdots, N\}$ with $u \neq v$
Move $n_{1u}$ in sequence $S_1$ to $n_{2v}$ in sequence $S_2$
Shift transactions $\in (u, v]$ in $S_1$ towards $u$ in $S_2$
```

Algorithm 3 Evolutionary sequence improvement

```
Choose sequence $S_1$ and determine $U_{\text{total}}(S_1)$
for $i = 1$ to $N_I$ do
  $S_2$ = mutate($S_1$)
  Determine $U_{\text{total}}(S_2)$
  if $U_{\text{total}}(S_2) > U_{\text{total}}(S_1)$ then
    $S_1 = S_2$
    $U_{\text{total}}(S_1) = U_{\text{total}}(S_2)$
  end if
end for
return $S_1, U_{\text{total}}(S_1)$
```

The result of this step is a sequence with optimized sum utility. As you can see in Algorithm 1, important input variables are the estimations for the CQI $c_j(t)$ and the remaining bits l_j for all transactions $j \in \{1, \ldots, N\}$. While determining the scheduling sequence, CI allows to plan the scheduling some time into the future. Such anticipatory scheduling accounts for the actual and predicted channel conditions as well as for the assumed transaction sizes. The sequence improvement is performed in each TTI. Naturally, if the set of active transactions has not changed, we reuse the previous sequence as the starting sequence in the next TTI. This improves the convergence behavior and reduces the required number of iterations N_I. When new transactions arrive, we increase the number of iterations by a scaling factor to account for the change in the set of transactions.

Step 2: combination with PF scheduling

Step 2 combines the CARA sequence from step 1 with a PF scheduling weight. This increases the channel-awareness of the heuristic. As a consequence, resource allocation may deviate from the sequence from step 1 if justified by the channel-state, e.g., when channel conditions for the first transaction in the sequence are poor but later transactions have a good channel.

The PF scheduling weight w_n of transaction n is determined as follows

$$w_n = \frac{c_n}{\overline{c_n}} \tag{10}$$

where $\overline{c_n}$ is the moving average of the channel quality. It is updated in each TTI with c_n if n is scheduled, and with 0 otherwise. As typical for PF, this weight compares the current channel state of a user to its average channel state and increases, when the user is not scheduled in a TTI to provide fairness.

Then, the combined weight v_n of transaction n_{1j} at position j in the sequence S_1 is calculated as follows

$$\forall j: \; v_{n_{1j}} = w_{n_{1j}} - p \cdot (j - 1) \tag{11}$$

where $p \in [\,0, \infty)$ is called penalty-factor. Thus, the weight penalty of a transaction increases with its position in the list. The free parameter p trades off the influence of the CARA sequence versus the PF weight. Note that p allows operators to decide whether to focus only on maximum utility or rather on the total system throughput and PF.

Finally, the transaction n^* with the largest weight is scheduled

$$n^* = \arg\max_n v_n. \tag{12}$$

Algorithm complexity

In comparison to solving the optimization problem (5) to (9) by linear programming, our heuristic reduces the complexity by restricting itself to the order of transactions. The complexity of the heuristic depends on two factors. First, the number of iterations N_I of the evolutionary method is a free parameter that directly defines the computational complexity of Algorithm 3. We use $10 \cdot N_I$ iterations whenever a new transaction arrives at the scheduler or the requirements of a transaction change. This improves the convergence by allowing more iterations to find a place for the new or modified transaction, which may change the whole sequence order. When nothing has changed, we use the predefined number of iterations N_I, because we can benefit from reusing the previous sequence in this case.

Second, the number of active transactions N affects the complexity, since we need to estimate all transaction finish times and aggregate their utilities in each iteration (Algorithm 1).

Consequently, we get an overall complexity in the order of $O(N \cdot N_I)$. Please note that, typically, $N_I \geq N$. This means that the complexity order is at least $O(N^2)$.

System model

Scenario

The studied scenario is presented in Figure 3. We focus on the downlink of a single wireless cell that serves 20 UEs. To account for interference, 6 permanently transmitting base stations are placed around the evaluated cell. Each of the base stations transmits at constant power, as given in Table 1.

The radio and physical layer (PHY) models represent a typical LTE system with frequency division duplexing. Spatial propagation is simplified, as base stations as well as UEs are equipped with isotropic antennas and the UEs are i.i.d. uniformly placed. A user's position does not change during one simulation run. Instead, we run multiple independent replications to get a uniform distribution of UEs over the simulated area. Due to shadowing, the wireless channel varies at a time scale of seconds. Additionally, the channel varies at a time scale of milliseconds due to fast fading. The channel coherence time is 3.4 ms. Link adaptation, i.e. modulation and coding scheme, is idealized by Shannon's equation that is clipped once the resulting PHY rate exceeds the threshold in Table 1. At this threshold, the PHY employs its largest modulation order and least robust channel code and cannot further increase the bitrate. In this idealized link layer model, we neglect retransmissions.

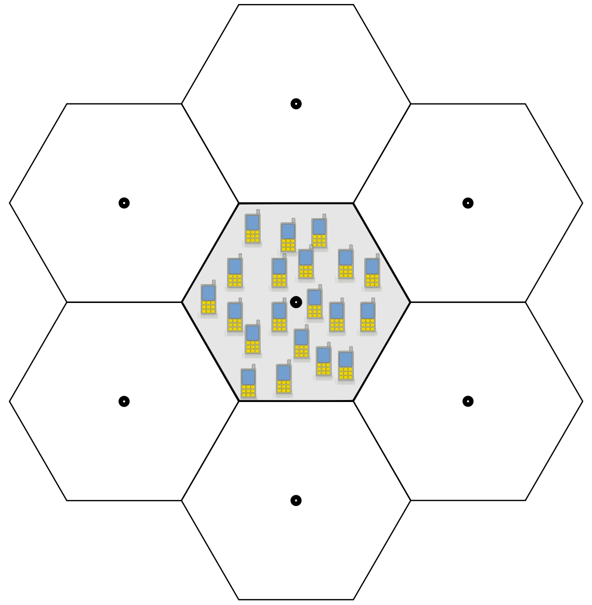

Figure 3 Simulation scenario; hexagonal arrangement of base stations, evaluated UEs are placed in the center cell.

Table 1 System model parameters

Property	Value
Cellular layout	hexagonal, 7 sites
UEs per cell	20
Inter BS distance	1 km
BS/UE height	32 m / 1.5 m
Carrier frequency	2 GHz
System bandwidth	10 MHz
BS TX power	46 dBm
Antenna model	Isotropic
Path loss	$128.1 + 37.6 \log_{10}(d)$, distance d in km [16]
Shadowing	8 dB log-normal, correlation distance 50 m
Multipath propagation	Rayleigh fading with Jakes-like temporal correlation [17],
	frequency-selective fading with Vehicular A channel taps [18]
UE velocity	10 km/h for shadowing and multipath propagation
Frame duration	1 ms
Link adaptation	Shannon's equation with SINR clipping at 20 dB

To isolate the effect of scheduling, we ignore further control loops at the link layer and higher layers, such as TCP. The radio resource management equally distributes the transmit power to all frequency bands and allocates the full bandwidth to a single UE. While equal power distribution is typical for LTE, ignoring subband allocation leads to pessimistic results but, again, allows us to isolate the gains of the proposed schedulers. For comparison, we complement the results by the ergodic channel capacity that provides the maximum average data rate for a system with ideal power and subband allocation. All studied schedulers operate on a TTI of 1 ms and with a separate queue per transaction. Once per TTI, the base station's scheduler decides which transmit queue receives the channel resources.

Traffic model

Our traffic models are based on specifications of cellular operators [19]. We focus on two common types of best effort traffic: Web surfing via HTTP and file downloads via the File Transfer Protocol (FTP) protocol. The object size of the FTP class has a truncated lognormal distribution, with the parameters given in Table 2, while the HTTP model details the transmitted elements. Each transmitted web page consists of the main object (i.e., HTML text) and a random number of embedded objects (e.g., pictures and linked JavaScript). Per page, the number of the embedded objects follows a truncated

Table 2 Parameters for traffic model distributions[19]

Object type	μ	σ	Minimum (Bytes)	Maximum (MBytes)
FTP	14.45	0.35	0	5
HTTP main obj.	8.37	1.37	100	2
HTTP embedded obj.	6.17	2.36	50	2

Pareto distribution and the size of all objects independently follows truncated lognormal distributions. According to [19], we parametrized the Pareto distribution with a mean of 5.64 and a maximum of 53. The remaining parameters are given in Table 2. We model the Inter-Arrival Time (IAT) of these objects at the transmission queue as i.i.d. negative exponential process. We vary the IATs to model the different offered loads on the system. The number of transactions per user is not limited. The latter assumptions reflects modern Smartphones, where multiple applications generate multiple requests in parallel.

In the following study, we assume a traffic mix of 90% HTTP transactions and 10% FTP transactions. This corresponds to 20% and 80% of the data volume, respectively. Furthermore, we vary this traffic mix when studying utility and complement our evaluation by results for pure FTP traffic.

Utility functions

For all schedulers, we employ the utility functions from Section "From application layer information to utility functions" to evaluate the QoS. We choose the parameterization of these functions as given in Table 3. The data rate the users expect for HTTP transactions is chosen to be $d_{\text{exp,HTTP}} = 6\,\text{MBit/s}$. Unlike HTTP transactions, file downloads are modeled as a background task, where the user expects a lower rate of $d_{\text{exp,FTP}} = 3\,\text{MBit/s}$. As in overloaded scenarios some transmissions cannot be completed in limited time, transmissions are dropped when their utility falls below a threshold of $U_{\text{drop}} = 0.01$. Dropped transactions have a utility of $U = 0$.

We obtain the parameter x, defining the inflection point of the utility curve, by solving (2) in x as $x = \frac{t_{\text{infl}} - t_{\text{start}}}{t_{\text{exp}} - t_{\text{start}}}$. We parametrize x according to the Mean Opinion Score (MOS) for different waiting times from [20], where the authors studied the satisfaction of users when they have to wait for different applications. We map a MOS 4 of satisfied users to $U(t_{\text{exp}})$ and a MOS 2 of dissatisfied users to $U(t_{\text{infl}})$. We then calculate the ratio of the waiting times for the application classes "Web Site" and "Download". The resulting values for HTTP and FTP traffic are given in Table 3.

Table 3 Parameters for utility functions

Parameter	HTTP	FTP
$U(t_{\text{exp}})$	0.95	0.95
d_{exp}	6 MBit/s	3 MBit/s
x	5.4462	5.7799
U_{drop}	0.01	0.01

Note that this parameterization of the utility functions is an example. Nonetheless, we observed that our algorithms yield similar results for other parameterizations of the utility functions. Figure 4 shows the results for different choices of the user expectations d_{exp}.

Performance evaluation

Evaluated schedulers and parameterization

In this section, we compare the performance of the CARA heuristic to the commonly applied PF scheduler [15] and to a Weighted Proportional Fair (WPF) scheduler, which represents QoS differentiation by traffic class. WPF simply enhances the instantaneous scheduling weight of ordinary PF with a scaling factor. If not mentioned otherwise, we use weight 2 for HTTP traffic and weight 1 for FTP traffic.

The LTE standard allows a differentiation by bearer for traffic prioritization. This allows increasing the QoS of interactive applications with lower latency requirement compared to background transmissions. As we show in the following sections, the CARA approach offers superior performance compared to WPF because it exploits more detailed CI and directly improves the user-observable metrics. Furthermore, CARA is also able to consider intra-class differentiation based on the applications' actual latency requirements, which is not possible with differentiation by bearer.

For CARA, the considered CI investigated in our performance evaluation are transaction length, transaction-individual latency requirements, and channel information. In our studies, we evaluate mainly two parameterizations of the CARA heuristic. The case "CARA-Heuristic with PF" is our heuristic from Section "Practical scheduling heuristic" with a penalty parameter of $p = 1$, while "Strict CARA-Sequence" is the same heuristic with the penalty parameter set to $p = 1000$. This high p practically avoids any deviation from the determined transaction sequence. Furthermore, we also show results in dependence of a varying penalty parameter.

The performance results are studied for increasing average load. Load is increased by decreasing the IAT between the traffic objects. The error bars in all plots are given for a 95% confidence level. The simulation results were obtained by 20 independent replications with a different choice of user placements and channel variations. Each replication has a warm-up-phase of 600s and a simulation phase of 600s.

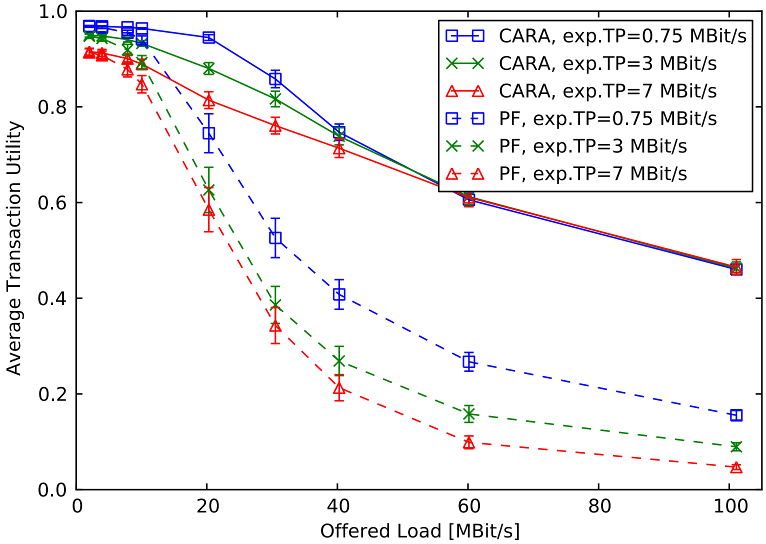

Figure 4 Average transaction utilities for different offered load and different expected throughput (TP) by the users.

Comparison of different scheduling algorithms

Figure 5 shows the average transaction utilities of different scheduling algorithms versus the offered load. Here, the average utility of the PF reference drops sharply as the offered load increases. This is because PF is not aware of transaction requirements. Hence, users do not get their expected data rate when the system is highly loaded. Consequently, a large fraction of the transactions miss their deadline which decreases utility.

WPF and CARA improve the utility compared to PF, because they are aware of the users' requirements and can use this knowledge for the scheduling decision. But while WPF reduces the flexibility to consider the channel states, the CARA heuristic commonly evaluates channel states and CI. The strict CARA sequence considers the channel during sequence determination, only. Additionally, the CARA heuristic with PF also verifies in each TTI, if the current CQI is in favor of the allotted transaction. This results in a high performance gain of our CARA heuristic. We compare the schedulers at an average utility of $\approx$ 0.86. This is an operation point of the network where most users are satisfied, so it could be chosen in a real network. At this utility, PF can handle an offered load of 10 MBit/s while CARA achieves up to $\approx$ 30 MBit/s. For

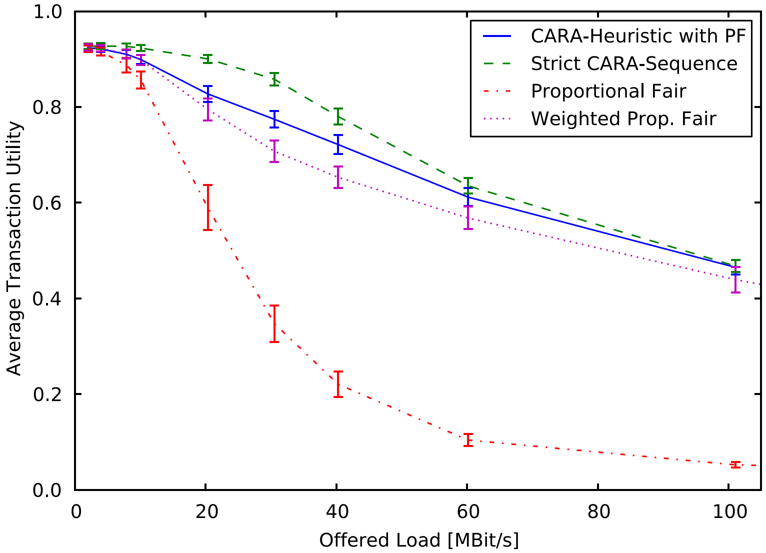

Figure 5 Average achieved transaction utilities for different offered load and scheduling algorithms.

the strict sequence heuristic, this triples the supported load compared to PF while maintaining the same average utility and without requiring additional channel resources. Compared to WPF, it is almost a factor of two. CARA with PF can still handle about twice the load of PF and is slightly superior to WPF. We expect that with more diverse traffic mixes and application requirements, the performance advantage of CARA over WPF is likely to increase, as the differentiation by traffic-class is very coarse.

Figure 6 shows the cell throughput. This metric is upper-bounded by the ergodic cell capacity, which can be calculated via Waterfilling and max-rate scheduling [21, Sec. 5.4.6]. Under the above channel and PHY assumptions, the ergodic cell capacity is 58.2 MBit/s.

Clearly, this upper limit cannot be achieved when schedulers have to account for fairness and when the channel is not fully loaded. This is shown in Figure 6, where all studied schedulers achieve low cell throughput when the offered load is low. Until 30 MBit/s offered load, the cell throughput increases, as more channel resources are used and multi-user diversity can be exploited. For higher loads, the cell throughput saturates at nearly half the cell capacity. This limitation results from the fairness constraints of the studied schedulers and from the traffic statistics which allow that users have nothing to transmit at some point in time.

Comparing the throughput results in Figure 6 shows that PF alone and CARA with PF outperform the other schedulers. This is not surprising as PF maximizes the rate at the cost of delay and, thus, decreases utility.

WPF has a higher throughput than the strict CARA sequence, because it focuses on the rate within each class and allows to schedule a transaction from the lower priority class, if there is a throughput advantage by a factor of two. In contrast, the strict CARA sequence focuses on maximizing utility only. It would only avoid bad channel states, if this leads to a lower overall utility.

If the load increases beyond 40 MBit/s, the throughput of the strict CARA sequence slightly decreases. This is because the scheduler can choose among many queued transactions and prefers those with a high achievable utility. This behavior demonstrates the influence of the transaction sequence which overrides the CQI-based decision. In contrast, choosing a lower p value (i.e., “CARA-Heuristic with PF”) decreases the priority of utility and closely achieves the cell throughput of PF in a heavily loaded system.

Let us now study CARA’s performance for a varying traffic mix. Figure 7 shows the influence of the fraction of HTTP transactions in the traffic mix on the utility performance of the schedulers. We choose an IAT such that the offered load is fixed to 30 MBit/s. Clearly, the utility results of our CARA heuristic are superior to PF over the whole range. However, CARA achieves the highest utility at an HTTP fraction of 0.9, containing both, FTP and HTTP traffic. This is because it offers an additional degree of freedom to trade-off the requirements of both traffic types. In this situation, PF lacks performance because it cannot distinguish between FTP and HTTP. Thus, it delays interactive HTTP transactions by scheduling FTP transactions.

Figures 8 and 9 show the average transaction utility and cell throughput for pure FTP traffic. Compared to our previous traffic mix, FTP leads to longer transactions and needs less transactions to fully occupy the wireless channel. With fewer active transactions in the system and less

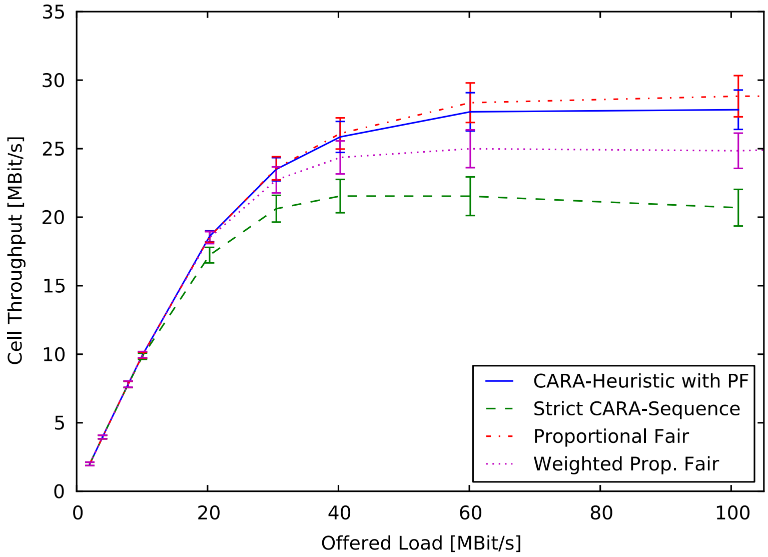

Figure 6 **Cell throughput for different offered load and scheduling algorithms.**

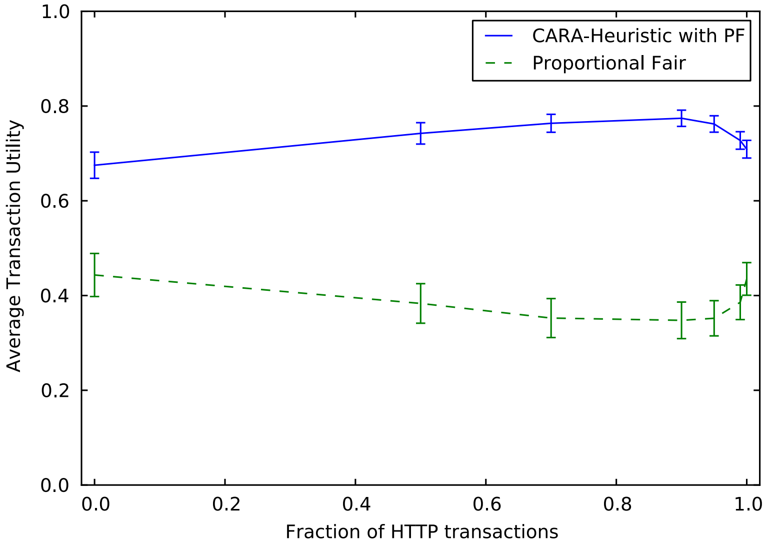

Figure 7 Average achieved transaction utilities for different fractions of HTTP traffic.

arrivals of new transactions, planning the resource allocation with the transaction sequence is more accurate and allows the CARA schedulers to get closer to ergodic cell capacity than with the above traffic mix. Furthermore, FTP transactions with relaxed latency requirements can be shifted in time to increase the flexibility for multi-user diversity and channel-awareness [15].

This improvement is clearly shown by comparing Figure 6 to Figure 9. Here, the cell throughput of both CARA cases strongly increases when the traffic changes to FTP. While the throughput of PF saturates at high load, CARA's throughput continues to increase. With pure FTP traffic, CARA outperforms PF since relaxed delay constraints allow to schedule transactions at later points in time. Consequently, CARA can shift transactions to time slots when the wireless channel of a UE is expected to improve. PF does not support such anticipatory scheduling at the time-axis.

Comparing both CARA heuristics shows that the strict CARA sequence achieves higher cell throughput than the CARA heuristic combined with PF. This results from the smaller number of active transactions and the slower change of the set of active transactions in comparison to the previous traffic mix. Therefore, a found sequence is stable for a longer time and transactions are scheduled when they are likely to have a good channel. In this regime,

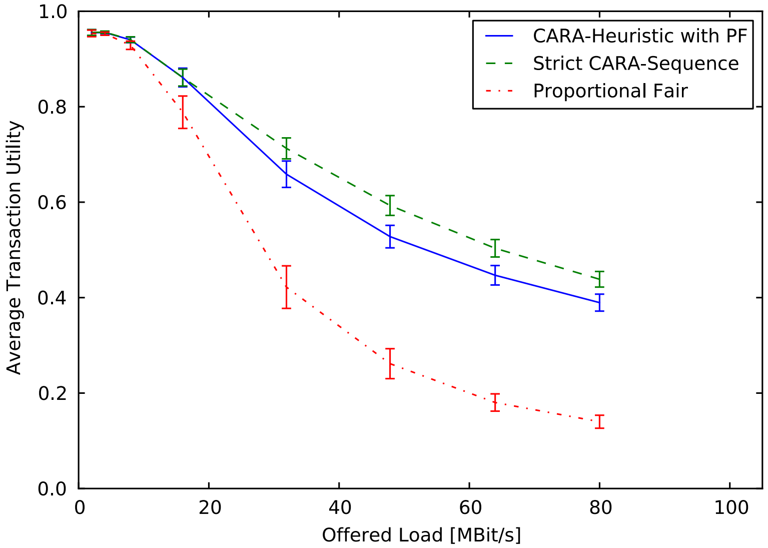

Figure 8 Average achieved transaction utilities for FTP-traffic only.

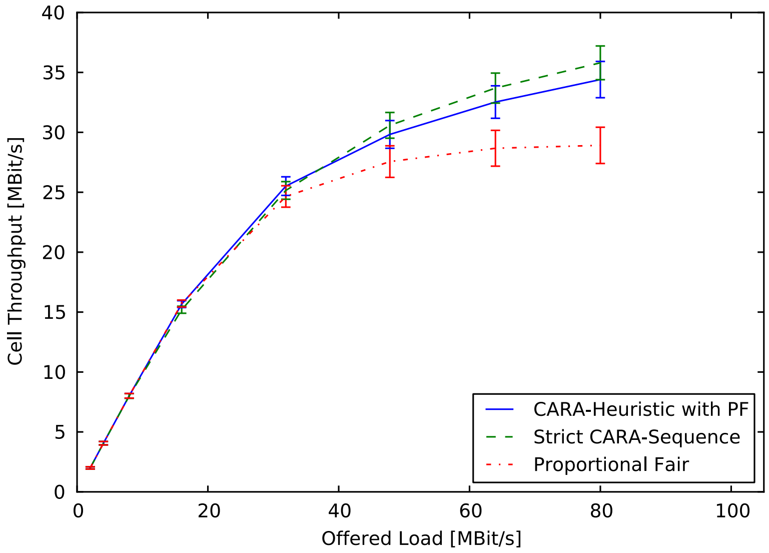

Figure 9 Cell throughput for FTP traffic only.

PF's reaction to short-term fluctuations of the channel state only disturbs the planned sequence and diminishes cell throughput.

Furthermore, we investigated different bandwidth demands from the users and different fractions of the total traffic controlled by CARA. Figure 4 shows utility results for different throughput demands of the users in the original scenario. While the utility is generally higher when user demands are lower, the curve's shape as well as the relation between CARA and PF scheduling remains similar to Figure 5. Together, these results attest a robust behavior under various traffic and system assumptions.

Trade-Off between utility and cell throughput

Figure 10 shows the resulting utility versus the cell throughput. Again, we obtained the results with an offered load of 30 MBit/s and the traffic mix described in Section "Traffic model". As before, we include the results for PF and WPF as references. Where PF aims for high cell throughput at the cost of utility, WPF increases average utility at the cost of cell throughput. However, CARA outperforms both reference schedulers.

As shown, this trade-off between utility and cell throughput can be directly adjusted by the penalty parameter of the CARA heuristic introduced in (11). A penalty

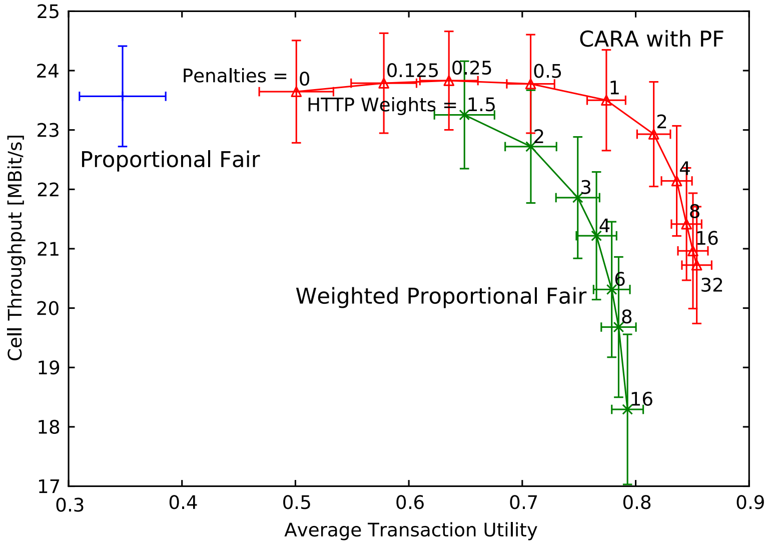

Figure 10 Influence of the Penalty-Parameter of the CARA heuristic for an offered load of 30 MBit/s. Trade-off between cell throughput and average utility.

of $p = 0$ means that the CARA sequence is completely ignored. CARA still outperforms PF, as it benefits from the assumption that the channel is known for the prediction time. When a new transaction arrives, it initializes the moving average with the average CQI calculated over the prediction window. In contrast, PF does not save the channel state of inactive users and the moving average has to be initialized with the current CQI, when an inactive user becomes active again.

A penalty of $p \geq 16$ practically ignores the PF scheduling weight and focuses only on utility maximization. Then, a part of the cell throughput is sacrificed for utility. Small changes in p do not lead to significant or unexpected changes in cell throughput and utility. This demonstrates that our CARA heuristic is robust with respect to the parameter p. Practically, one would not choose $p < 1$ as such values do always decrease utility without increasing cell throughput. The Pareto optimum with respect to utility and cell throughput is achieved for $1 \leq p \leq 4$.

Weighted proportional fair can offer a similar trade-off by adjusting the weight of the HTTP-class. In Figure 10, we varied the HTTP-weight between 1.5 and 16, while giving FTP-transactions always a weight of 1. Clearly, WPF achieves neither the throughput nor the utility performance of the CARA-heuristic.

Increasing the penalty parameter also affects the fairness. We evaluate the fairness using Jain's fairness index [22]. For $p = 0$, i.e., pure PF scheduling, Jain's index is 0.93. We found that this index remains almost constant for $p \leq 1$. It slightly decreases until 0.89 is reached for very large p. At such a high penalty, PF does not contribute anymore to the decision and it costs more resources to assure a high utility to users at the cell border.However, the results for the fairness index show that even the strict CARA sequence offers fairness close to that of the PF algorithm.

Performance with deteriorated CI

As mentioned in Section "Deployment issues", it cannot be assumed that all UEs are equipped with context signaling functionalities right away. Furthermore, there are always situations, when the required CI is not available, e.g. when the UE does not know the size of a transaction in advance. We investigated such deteriorations of CI with simulation studies of which we present two in the following. The studies use a fixed operation point of 30 MBit/s.

First, the exact size information of a transaction may be missing. In such a case, the scheduler assumes a size estimate depending on the traffic class. We model the inaccuracies of such estimations by adding a size error relative to the real size of a transaction to the scheduler assumptions. As our transaction sizes are distributed log-normally, we assume that also the error distribution follows a log-normal distribution. We obtain the transaction sizes L^* used by the scheduler from

$$L^* = L \cdot \exp^{\mathcal{N}(0,\sigma^2)} \tag{13}$$

With this, we estimate too small and too large transaction sizes in 50% of the cases, respectively. We then vary the severity of the size error by adapting the variance σ^2.

Figure 11 shows the results in terms of utility and Figure 12 shows how cell throughput is affected. The

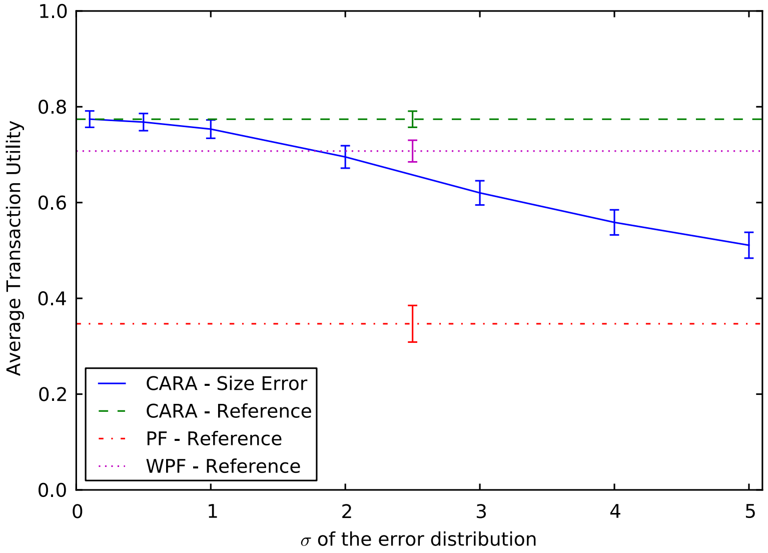

Figure 11 Utility degradation in dependence of the variance σ of the error distribution on the transaction size information.

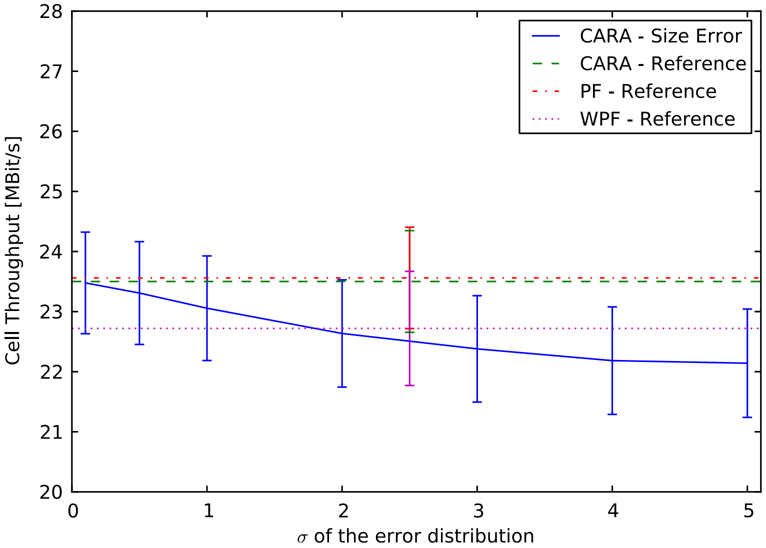

Figure 12 Cell throughput degradation in dependence of the variance σ **of the error distribution on the transaction size information.**

CARA scheduler is configured with $p = 1$ and the reference line represents ideal context signaling. For small error variations up to $\sigma = 1$, the average utility nearly stays constant. For larger variances, we get a slight utility degradation. CARA's utility performance always remains superior to PF and matches WPF at $\sigma \approx 2$. With inaccurate transaction size information, cell throughput starts declining right away, as can be seen in Figure 12. This is due to the fact that the planning of the transaction order is disturbed and transactions cannot be served at times when the respective UE has a good channel quality. Concluding from the results with deteriorated size information, we can say that it is sufficient for an efficient operation of the CARA heuristic to know the order of the size of transactions. Please note that for $\sigma = 2$, already about 30 % of the transaction sizes are wrong by one order of magnitude.

The second study of deteriorated CI investigates the effect of delayed context signaling. For larger delays this is as if the information would be missing completely, as the transaction is already finished or too late, when the information finally arrives.

When a transaction arrives at the base station for which no CI has been signaled, the scheduler has to fall back

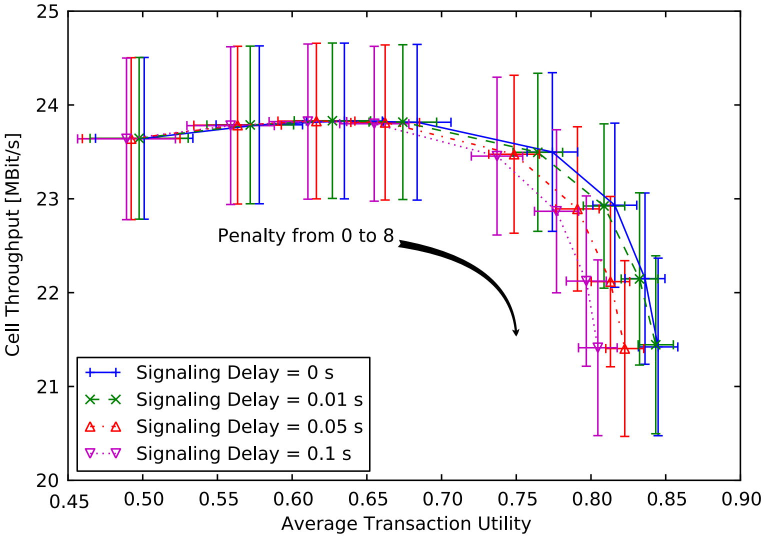

Figure 13 Reduction of the rate region of the CARA scheduler for delayed context signaling when assuming a default transaction similar to HTTP transmissions.

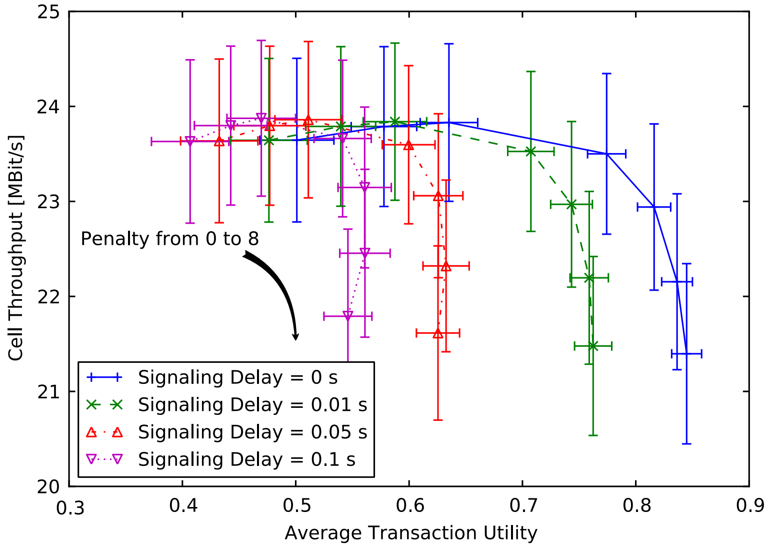

Figure 14 Reduction of the rate region of the CARA scheduler for delayed context signaling when assuming a default transaction similar to FTP transmissions.

to default values. We modeled such a behavior with two alternative default transactions:

- Default transaction 1: $L = 1$ MBit; $x = 5.4462$ (similar to HTTP transactions)
- Default transaction 2: $L = 16$ MBit; $x = 5.7799$ (similar to FTP transactions)

To test the worst case behavior, the scheduler either always uses default transaction 1 or default transaction 2. Figures 13 and 14 show the respective results for the rate region of utility and cell throughput with varying signaling delays. For the investigated traffic mix, 90% of the real transactions are similar in size and requirements to default transaction 1. Therefore, the performance of CARA only slightly degrades even for delays up to 100 TTIs. Incoming FTP transactions are served erroneously as HTTP transactions in the beginning, but this is not as severe as the other way. In Figure 14, we show the result from using default transaction 2, similar to FTP transactions. The utility strongly degrades if signaling delay increases, especially for a large p. Not only is the default transaction quite different from most transactions, also transactions without CI at the scheduler are usually postponed until the signaling arrives. This means that many

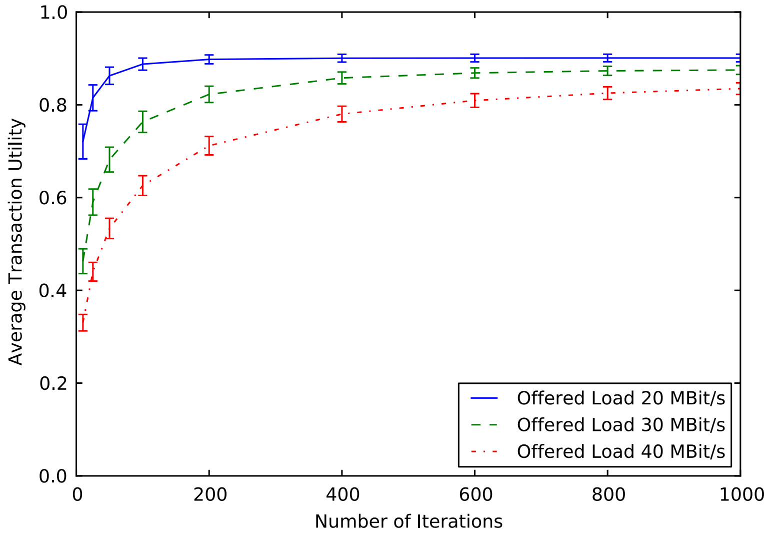

Figure 15 Utility gains in comparison to the invested computational complexity for the CARA-Heuristic with PF.

transactions already missed their expected finish times, when signaling arrives.

Concluding from these results with delayed signaling, we can say that by using reasonable default values, e.g. derived from the application type, the impairments on CARA's performance can be kept at a low level.

Additionally to the presented studies, we investigated deteriorations of latency requirements and channel prediction which showed only smooth degradations and attest CARA a robust behavior against deteriorated CI.

Scheduling complexity and performance improvement

By adjusting the number of iterations per TTI N_I, we can control the computational complexity our heuristic adds to the base station. As we see in Figure 15, the utility already quickly increases with a small number of iterations. For higher N_I, the utility saturates. This is of particular interest in the low load regime (i.e., offered load below 20 MBit/s), where the maximum average utility can already be achieved with 100 iterations or less. More iterations are needed for higher load, since the number of active transactions increases. This leads to more possibilities for the order of the transaction sequence and, thus, requires a larger N_I until the sequence converges.

Apart from such additional iterations, increasing the offered load leads to a higher computational complexity of the algorithm itself. First, the utility determination includes more transactions to be summed up. Second, with more frequent transaction arrivals we cannot reuse the previous sequence so often which increases the number of iterations again. From Figure 15, we see that for an offered load of 30 MBit/s the algorithm converges at about 400 iterations. At this operation point, we have 50 active transactions on average.

Conclusion

We demonstrated that exploiting CI at the base station's wireless scheduler is worth the effort. Our context-aware scheduler converges quickly and substantially improves the users' QoS or, alternatively, increases the supported traffic load. We found that CARA triples the supported load per cell in various traffic scenarios, when compared to PF scheduling. This high gain is achieved without decreasing the QoS and without demanding for more channel resources.

Context-aware resource allocation's high gain is based on a transaction framework that informs the scheduler about the application's flows and related delay requirements. Consequently, the scheduler can (i) allocate future time slots to flows instead of single packets and (ii) account for the individual delay budget of each application by using time-utility functions. The result is a scheduler that performs a throughput-delay tradeoff and is efficient in terms of spectrum and computation time. We demonstrated that CARA is beneficial even with limited CI and adds only little signaling overhead to the access network traffic.

We conclude that CARA is a powerful and practical approach to cope with the intense traffic requirements of modern UEs. Its generality allows easy integration of further CI such as mobility parameters. Current studies on employing user trajectories and application requests for inter-cell scheduling [23] show significant gains. This indicates once more that CARA is a promising field of future research.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors thank Christian M. Mueller (IKR) and Michael Timmers (Bell Labs, Antwerp) for their helpful comments. This work was supported by Bell Labs, Stuttgart and by the DFG program Open Access Publishing.

Author details

[1] Institute of Communication Networks and Computer Engineering, Universität Stuttgart, Stuttgart, Germany. [2] Bell Labs, Alcatel-Lucent, Stuttgart, Germany.

Received: 12 October 2011 Accepted: 4 June 2012
Published: 12 July 2012

References

1. U Paul, AP Subramanian, MM Buddhikot, SR Das, in *Proceedings of the IEEE INFOCOM,* Understanding traffic dynamics in cellular data networks (ACM, 2011), pp. 882–890
2. G Maier, F Schneider, A Feldmann, in *Proceedings of the 11th International Conference on Passive and Active Network Measurement (PAM), Volume 6032,* A First look at mobile hand-held device traffic, (2010), pp. 161–170
3. H Falaki, R Mahajan, S Kandula, D Lymberopoulos, R Govindan, D Estrin, in *Proceedings of the 8th international conference on Mobile systems, applications, and services,* Diversity in smartphone usage (ACM, 2010), pp. 179–194
4. W Leland, M Taqqu, W Willinger, D Wilson, On the self-similar nature of ethernet traffic (extended version), IEEE/ACM Trans. Netw. **2**, 1–15 (1994)
5. S Ryu, B Ryu, H Seo, M Shin, in *Proceedings of the IEEE International Conference on Communications (ICC),* Urgency and efficiency based packet scheduling algorithm for ofdma wireless system, vol 4, (2005), pp. 2779–2785
6. X Xiao, L Ni, Internet QoS: a big picture. IEEE Netw. **13**(2), 8–18 (1999)
7. G Song, Y Li, G Song, Y Li, Utility-based resource allocation and scheduling in OFDM-based wireless broadband networks. IEEE Commun. Mag. **43**(12), 127–134 (2005)
8. F Kelly, Charging and rate control for elastic traffic. Eur. Trans. Telecommun. **8**, 33–37 (1997)
9. WH Kuo, W Liao, Utility-based radio resource allocation for QoS traffic in wireless networks. IEEE Trans. Wirel. Commun. **7**(7), 2714–2722 (2008)
10. JW Lee, JA Kwon, Utility-based power allocation for multiclass wireless systems. IEEE Trans. Veh. Technol. **58**(7), 3813–3819 (2009)
11. G Maier, A Feldmann, V Paxson, M Allman, in *Proceedings of the 9th ACM SIGCOMM Internet measurement conference,* On dominant characteristics of residential broadband internet traffic, (2009)
12. J Nielsen, *Website Response Times*(2010). http://www.useit.com/alertbox/response-times.html. Accessed on 24 Aug 2011
13. RB Miller, in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I,* Response time in man-computer conversational transactions, (1968)
14. M Proebster, M Kaschub, S Valentin, in *Proceedings of the IEEE International Conference on Communications (ICC),* Context-aware resource allocation to improve the quality of service of heterogeneous traffic (ACM, USA, 2011), pp. 1–6

15. P Viswanath, D Tse, R Laroia, Opportunistic beamforming using dumb antennas. IEEE Trans. Inf. Theory. **48**(6), 1277–1294 (2002)
16. *Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA)*. TR 25.814, 3GPP WSG RAN 2006
17. P Dent, G Bottomley, T Croft, Jakes fading model revisited. Electron. Lett. **29**(13), 1162–1163 (1993)
18. *Selection procedures for the choice of radio transmission technologies of the UMTS*. TR101112/UMTS30.03 V3.2, ETSI 1998
19. (R Irmer, ed.), *Radio Access Performance Evaluation Methodology*. NGMN White Paper V1.3 2008
20. S Niida, S Uemura, H Nakamura, Mobile services. IEEE Veh. Technol. Mag. **5**(3), 61–67 (2010)
21. D Tse, P Viswanath, *Fundamentals of Wireless Communication*. (Cambridge University Press, Cambridge, 2005)
22. RK Jain, DMW Chiu, WR Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems*. TR-301, DEC 1984
23. H Abou-zeid, S Valentin, H Hassanein, in *Proceedings of Capacity Sharing Workshop,* Context-aware resource allocation for media streaming: exploiting mobility and application-layer predictions, (2011)

doi:10.1186/1687-1499-2012-216
Cite this article as: Proebster *et al.*: Context-aware resource allocation for cellular wireless networks. *EURASIP Journal on Wireless Communications and Networking* 2012 **2012**:216.