Context-Aware Resource Allocation to Improve the Quality of Service of Heterogeneous Traffic

Magnus Proebster*, Matthias Kaschub*, and Stefan Valentin†
*Universität Stuttgart, Institute of Communication Networks and Computer Engineering, Stuttgart, Germany,
magnus.proebster@ikr.uni-stuttgart.de, matthias.kaschub@ikr.uni-stuttgart.de
†Bell Laboratories, Alcatel-Lucent, Germany, stefan.valentin@alcatel-lucent.com

***Abstract*—By supporting advanced multimedia applications and multitasking, recent smartphone operation systems offer great functionality but make these devices even more difficult to serve. Now, network operators face a situation where many applications generate highly heterogeneous and bursty traffic patterns. To cope with this challenge, we introduce *Context-Aware Resource Allocation (CARA)*. This new approach enables a scheduler to exploit new information from the users' environment, thereby, allocating its resources more efficiently. We demonstrate CARA's high performance and feasibility by studying it as network utility optimization problem. Comparing our results to traditional proportional fair scheduling clearly shows that knowing more about a user's context is worth the effort.**

I. Introduction

Today, users request high throughput and real-time transmission even in cellular networks. This tremendous demand mainly results from the increased capabilities of current User Equipments (UEs) and from the range of applications that use these capabilities. Supporting multitasking, the latest generation of smartphones allows to transmit and to receive multiple traffic flows at once, thus, providing smooth operation and high flexibility to the users.

However, serving such up-to-date UEs is a challenge for network operators. With modern applications, users can generate a heterogeneous traffic mix where data, voice, video, and other interactive services have to be delivered simultaneously. All these services compete for the same wireless resources and can, thus, put high load peaks and heavy congestion to the wireless link.

In this paper, we cope with such heterogeneous load situations by *CARA*. This new approach combines two techniques:

- *Context awareness* informs the scheduler about the applications' environment. By receiving various context information from the UEs, the base station's scheduler can be aware of many context *features*. In this paper, we focus on the application's foreground/background state. Assuming that the scheduler knows if the traffic is generated by an application in the foreground or in the background of the UE's operating system, allows us to demonstrate that exploiting new context information is worth the effort.
- *Transaction-based scheduling* handles the additional complexity added by context awareness. This is done by managing traffic flows as *transactions* – each with a finish time, Quality of Service (QoS) requirements, and context information attached. Transaction-based scheduling provides the framework to organize the information flow and to build efficient schedulers. Unlike previous approaches, such a scheduler directly accounts for the transactions' finish times to meet real-time requirements. By predicting the size and the requirements of a transaction, the scheduler can extend its allocation decision to multiple Transmission Time Intervals (TTIs) in advance.

We will demonstrate that, by joining both techniques, CARA efficiently exploits new context information at feasible complexity. We do so by formulating CARA as Network Utility Maximization (NUM) problem. Unlike previous work [1]–[3], our analysis is based on time-variant network utility functions. That is, rather than formulating network utility as a function of rate, we use functions of TTI. By directly accounting for finish times, this new analytical approach allows us to study the trade-off between different traffic requirements.

Our paper is structured as follows. In Sec. II, we compare our approach to the related work. Sec. III describes the system model and the notation. In Sec. IV, we briefly recapitulate Proportional Fair (PF) scheduling that is a foundation of our approach. Sec. V presents our transaction-based scheduling framework. Here, we formulate our approach as optimization problem and discuss its complexity. After that, in Sec. VI we compare our approach to PF scheduling by simulation. Finally, we conclude the paper and discuss future work.

II. Related Work

Channel-aware scheduling has become a wide field of research. Here, PF scheduling [4] is a standard approach. Due to its high practical relevance, PF is the baseline of our following studies and will be further discussed in Sec. IV. Many current approaches include PF as a special case by incorporating QoS and fairness constraints in a NUM problem [1]–[3]. Comparing CARA to these approaches shows the following two major differences.

First, CARA allows the scheduler to exploit any type of information. While most schedulers base their decision only on channel state information [4]–[6], some of them account for the users' QoS classes via Utility functions. While many approaches choose equal Utility functions for all users, e.g., [1], more recent approaches incorporate heterogeneous QoS classes by choosing the Utility function for each user individually [2], [3]. CARA, however, exploits more information at the

TABLE I
SYSTEM MODEL PARAMETERS

Property	Value
Cellular layout	19 sites, 3-sector with wraparound
Inter BS distance	1000 m
BS TX power	46 dB
BS/UE height	32 m / 1.5 m
Antenna model	3D, from [8], Table A.2.1.1-2
Shadowing	8 dB log-normal
Multipath propagation	Rayleigh (modified Jakes' model [9]), Vehicular A channel taps [10]
UE velocity	10 km/h
Carrier frequency	2 GHz
Frame duration	1 ms

scheduler than these previous approaches. One example is the background/foreground state of the application generating the traffic. This context feature has not been exploited for wireless resource allocation so far.

CARA's second major difference to previous work is its focus on deadlines. CARA is explicitly designed to minimize the users' waiting times. This is neither the case with previous work on PF nor on NUM. While PF-based approaches [4], [6] maximize the weighted rate directly, even QoS-aware Utility maximization is based on rate-dependent Utility functions [1]–[3]. No previous approach uses time-dependent Utility functions to directly account for the users' delay constraints. By doing so, CARA not only reflects the users' delay requirements directly but significantly differs from the current literature.

III. System Model

To compare our approach to PF scheduling, we use the following system model. We evaluate the downlink data transmissions of 10 active UEs. Each UE transfers exactly one transaction as specified below. We repeat each evaluation 100 times with independent radio channels and UE positions. Each realization is evaluated for 10 seconds.

Our model for radio propagation corresponds to [7]. UEs are placed uniformly in a multi-sector scenario as defined in Tab. I. Only the communication in one sector of the central base station is evaluated, the other sectors are producing interference under full buffer assumption. All UEs connect to the base station with the highest average signal to interference and noise ratio. While the evaluated channels suffer from frequency-selective fading, the interfering channels are assumed to be frequency-flat. Due to the short evaluated time span of 10 s, Shadowing is assumed to be constant. A UE's throughput corresponds to the Shannon capacity for 50 orthogonal 180 kHz sub carriers, clipped at 20 dB. This reflects a 10 MHz system according to the Long Term Evolution (LTE) standards of the 3rd Generation Partnership Project (3GPP).

For the evaluation, we create a synthetic traffic scenario. This scenario differentiates two classes of transactions: Foreground and background. The traffic mix contains 50% foreground traffic. As representative applications for our evaluation, we choose interactive web-browsing on modern web pages with many embedded objects as foreground traffic and file-downloads as background traffic. Consequently, we assume an average transaction size of 2 MBytes where background transactions are assumed to be five times larger than foreground transactions in average. With a log-normal size distribution, we derive the following transaction sizes in bits:

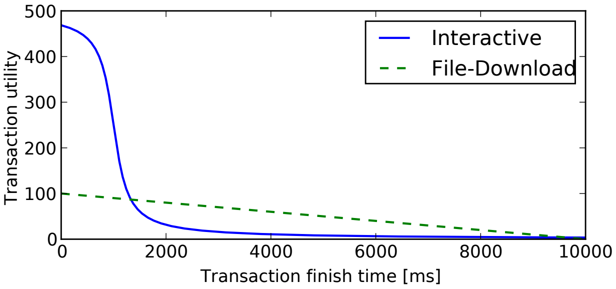

Fig. 1. Transaction utility $U_T(t)$ against transaction finish time for different traffic classes (typically foreground and typically background).

$$B_T = \begin{cases} 10^{\mathcal{N}(6.277,\,1)} & \text{foreground traffic} \\ 10^{\mathcal{N}(6.926,\,1)} & \text{background traffic} \end{cases} \tag{1}$$

The mean of the normal distribution $\mathcal{N}$ in the exponent is obtained by inserting the assumed average transaction sizes into the formula for the mean of the log-normal distribution.

The utility functions for the finish time t_f of foreground and background traffic are demonstrated in Fig. 1 and calculated from:

$$U_T\left(t_f\right) = \begin{cases} 500 \cdot \left(\frac{1}{2} - \frac{1}{\pi} \cdot \operatorname{atan}\left(\frac{t_f}{200} - 5\,\mathrm{ms}\right)\right) & \text{foreground} \\ 100 \cdot \left(1 - t_f \cdot \min\left(\frac{1400}{B_T}, 5\right)\right) & \text{background} \end{cases} \tag{2}$$

The following considerations lead to this exemplary choice of utility functions. An important feature of the function is its slope, because it denotes the penalty or gain when postponing or advancing the respective transaction. Foreground or interactive traffic should take no longer than one second to complete, otherwise the utility for the user drops sharply. This is expressed by $\operatorname{atan}(\cdot)$ in (2). Background traffic such as file-downloads, for example, has a more continuous utility decay over time. However, a user expects a small file to arrive earlier than a large file. Thus, the slope is proportional to the transaction size B_T down to a certain minimal slope. The scaling factors and offset parameters are chosen such that we obtain a reasonable weighting between foreground and background traffic. Hereby, the absolute parameter values have no further meaning than the shape and relative position of the utility functions.

IV. Proportional Fair Scheduling

Due to its high practical relevance, PF scheduling serves as a baseline for our context-aware resource allocation strategy in Sec. V and for our studies in Sec. VI. Before describing a practical PF scheduler, let us formally introduce proportional fairness and the underlying NUM problem.

To each user $n = 1, \ldots, N$ we allocate a vector of resources r_n. Our objective is to find the allocation $r = r_1, \ldots, r_N$ that maximizes the sum of the network utility function $U_n(r_n)$ over all users. Formally, this well-known NUM problem can be stated as

$$\begin{aligned} \text{maximize}_r \quad & \textstyle\sum_{n=1}^{N} U_n(r_n) \\ \text{subject to} \quad & \textstyle\sum_{n=1}^{N} r_n \le R, \quad 0 \le r_n \le R \; \forall n \end{aligned} \tag{3}$$

where R stands for the total amount of resources in the cell. We call an allocation r *feasible*, if it fulfills the constraints in (3) and we call a feasible allocation r *optimal*, if for any other feasible allocation $r', U(r) \geq U(r')$. We denote such an optimal allocation by r^*.

Any feasible resource allocation r is *proportionally fair* if for any other feasible allocation r' the aggregate change is zero or negative, i.e., $\sum_{n=1}^{N}(r'_n - r_n)/r_n \leq 0$. In [4] it was shown that the solution of (3) is proportionally fair if $U_n(r_n) = \log(r_n)$. Therewith, we incorporate proportional fairness into (3) by choosing a logarithmic utility function for each user. This strictly concave function and the linear sum constraint in (3) now allows to apply standard methods of convex optimization [11]. Typically, a Lagrangian decomposition of (3) is solved either by gradient projection or by a penalty algorithm [2].

Coupling proportional fairness to logarithmic utility functions becomes handy to derive a low-complexity heuristic. Such strategy was introduced in [5] by allocating all resources to the user n^* with largest weight $w_n = r_n/\bar{r}_n$. It was shown that this simple strategy solves (3) if $U_n(r_n) = \log(r_n)$ and if $\bar{r}_n$ is a long-term average. Hence, even an optimal PF solution is provided when $\bar{r}_n$ is perfectly known. In practice, however, $\bar{r}_n$ is a moving average over a finite time-window of duration T_c. Then, the above scheduling strategy provides suboptimal solutions that converge to proportional fairness [6].

V. Transaction-based, Context-aware Scheduling

A. Transaction Framework

Our goal is to bring context-information to resource allocation in order to increase the system-wide QoS. On the UE side, we have the application knowing about its network traffic requirements and the current situation it is running in. This information makes it possible to determine if an application is running in the foreground or in the background and allows a prioritization of the transmitted data. To transport the information to the base station, we use the notion of a transaction. A transaction contains all network activity that follows from a user action (or background operation) until the required content is shown. For example, a user clicks on a web link and waits to see the new content in his web browser. Then, the web page itself and all embedded objects belong to this transaction.

Information on a transaction not only contains QoS requirements. Also the size of the data needs to be signaled to the base station. In this paper we assume that the size of a transaction is perfectly known at the base station at no signaling cost. While this assumption is idealistic, it is not infeasible. Real systems can provide such estimates by inspecting packet headers of transport or application layer protocols, e.g., Real-Time Transport Protocol (RTP).

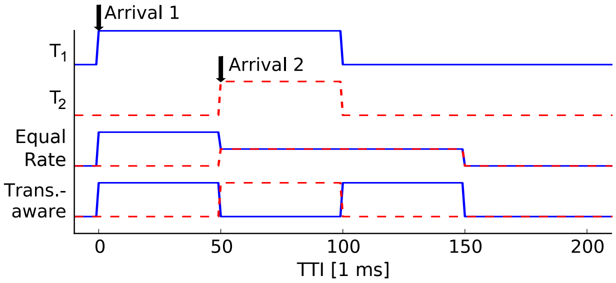

Fig. 2. Advantage from knowing remaining transaction sizes. Comparison between equal rate scheduling and transaction-aware scheduling.

Depending on the application, the QoS requirements may differ. For streaming applications, a certain data rate should be sustained. In our example of interactive web browsing, however, the reactivity and QoS is expressed best by the time required to finish a transaction. This finish time is a good metric to reflect the user satisfaction with his network connection [12]. While rate metrics only result in a certain finish-time over multiple TTIs, the transaction finish time does not require intermediate rate constraints but directly describes the Quality of Experience (QoE) for the user. We express this QoE by defining Transaction Utility Functions (TUF) and illustrate two examples in Fig. 1. These functions depend on transaction finish times and – since users expect their transmissions to finish in limited time – are monotonically decreasing. The faster such a TUF decreases, the stronger a quick response is valued for the user.

This new utility function allows us to consider context knowledge to plan resource allocation on a larger time scale instead of just for one TTI. By knowing the transaction sizes, it is possible to predict traffic for the near future. As depicted in Fig. 2, the scheduler can postpone longer transactions in favor of short transactions in order to improve the finish time of the short transaction without disturbing the long one. In this example, transaction T_1 has twice the amount of data of transaction T_2 to transmit. T_1 starts its transmission at $t = 0\,\text{s}$ and T_2 at $t = 50\,\text{s}$. The optimal choice for fair scheduling is equal rate. This means both transactions get half of the bandwidth and complete their transmission at $t = 150\,\text{s}$ (if we assume equal and constant channels). We assume that the scheduler knows the transaction sizes of both transactions. Thus, it serves T_2 first and then continues to serve T_1. While T_1 has the same finish time as with equal rate scheduling, T_2 completes its transmission at $t = 100\,\text{s}$. This simple example demonstrates the gain achievable with knowledge of transaction sizes.

Also, the differentiation between application requirements allows a higher scheduling flexibility. While file downloads or other background tasks can accept a certain delay, interactive applications in the foreground have stricter delay constraints. These different delay constraints allow us to choose the utility curves as in Fig. 1. Other applications like software updates or periodic tasks may again be represented by different utility functions. Consequently, urgent traffic can be scheduled more flexible by prioritizing it against non-urgent traffic.

The next step is to map transaction finish times to actual resource allocation. To demonstrate the advantages of our transaction framework, we formulate it as an optimization problem for a predefined duration (see Sec. V-B).

In contrast to that an implementable scheduler needs a transformation or heuristic for this task. While transactions usually last for several TTIs, the scheduling decision has to take place for each individual TTI. On the one hand, we have the TUF, which is a decreasing function over time. On the other hand, the actual resource allocation can be done with utility-fair scheduling algorithms (e. g. [1]). For this, an instantaneous utility function (IUF), which is an increasing function with respect to the user rate, is needed in each TTI. A transformation between the TUF and IUF considers traffic prediction and channel information in order to exploit the given context information. In effect, we will get IUFs which vary over time.

Summing up, the above framework has two benefits: By estimating the size of a transaction, the scheduler can prioritize short transactions. The second advantage is that it is possible to relax latency requirements in certain transactions where this is possible without reducing the QoS perceived by the user. With this, channel-dependent scheduling gets a larger degree of freedom which makes it possible to increase multi-user diversity [5], e. g. , by additionally exploiting channel-variations from shadowing.

Note that in cellular wireless systems, the base station has to memorize the state of active mobiles, e. g. , for mobility management and opportunistic scheduling. Therefore, the additional overhead introduced by buffering transaction information is small. Also the signaling effort, which has to be done only once for a transaction, is reasonable.

B. Optimization Problem

To realize resource allocation with the transaction-based framework, we formulate it as an optimization problem. The goal is to maximize

$$U_{total} = \sum_t \sum_T U_T(t) f_{T,t} \tag{4}$$

where U_T is the utility function of transaction T. This function depends on the transaction finish time t_f. $f_{T,t} \in \{0, 1\}$ is a "flag" which is one for the finish time $t = t_f$ of transaction T and zero for all other t. The transaction index is $T \in \{1, \dots, n\}$ for n transactions in total.

The decision variables of the above objective function represent the searched resource allocation. The resources $r_{T,t}$ are allocated to transaction T in TTI t and lead to a certain order of the finish flags $f_{T,t}$. This optimization problem has the following constraints:

$$\begin{aligned} \forall T:\ \textstyle\sum_t f_{T,t} &= 1 && (5)\\ \forall T,t:\ f_{T,t} &\leq \tfrac{1}{B_T}\left(\textstyle\sum_{t_1=1}^{t} r_{T,t_1}\gamma_{T,t_1}\right) && (6)\\ \forall t:\ R &\geq \textstyle\sum_T r_{T,t} && (7)\\ \forall T:\ B_T &= \textstyle\sum_t r_{T,t}\gamma_T(t) && (8)\\ \forall t < t_{0T}:\ r_{T,t} &= 0 && (9) \end{aligned}$$

Hereby, $\gamma_T(t)$ is the channel quality of transaction T at time t in bits/resource. B_T is the size of transaction T in bits. The time slots are $t \in \{1, \dots TTI\}$. t_{0T} is the start of a transaction and R is the total amount of resources.

Constraint (5) enforces that there is exactly one finish flag for each transaction. The finish flag is defined by (6) stating that a transaction can only be finished after all of its data has been transmitted. Hereby, the data transmitted in each TTI is calculated by the product of channel quality and allocated resources. With (7), we constrain the resource allocation to the available bandwidth. With (8), it is ensured that all transactions have to be finished. Finally, (9) introduces the possibility to model traffic with transactions that start at TTI t_{0T}.

The presented optimization problem implicitly leads to a channel-aware resource allocation. A high sum rate of the system by opportunistic scheduling leads to shorter transaction finish times and thus maximizes the optimization goal.

C. Algorithm Example

Solving the optimization problem as a Linear Program (LP) leads to high computational complexity. Now, the decision variables comprise the resource allocation of all transactions in each TTI. With n transactions and n_{TTI} TTIs, this leads to $O(n \cdot n_{TTI})$ decision variables. In addition to this high complexity order, the optimization problem is ill-conditioned such that typical LP frameworks cannot simplify it. This forces the solver to traverse the whole solution space which renders LP infeasible to solve problem (4) for reasonable numbers of transactions and time slots.

We reduce this complexity with two simplifications. First, setting $R = 1$ removes frequency selectivity from our scheduling problem. We chose this simplification since multiuser diversity gains due to frequency selectivity are well understood [5]. Second, we reduce the number of scheduling decisions by assigning resources to a transaction only every 20 TTIs. Note that this granularity still lies in the order of the channel coherence time for the evaluated scenario.

Nonetheless, even with these simplifications, problem (4) cannot be solved with LP on typical hardware. To this end, we change the optimization approach to a Genetic Algorithm (GA). This algorithm converges to the optimal solution without exploring the whole solution space. In this GA, each resource allocation $\{r_{T,t} : \forall T, t\}$ that solves our optimization problem (4) represents one *individual*. The fitness of different individuals is calculated as the sum of the transaction utilities. If a transaction does not terminate in a solution, the solution is not valid and the individual's fitness is zero. In each generation,

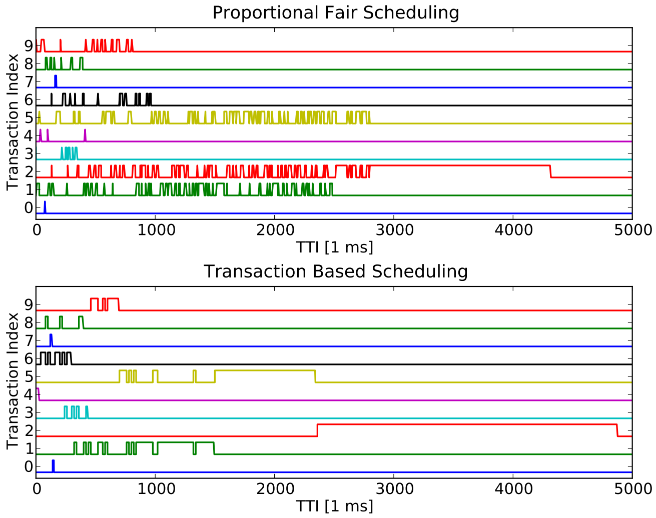

Fig. 3. Comparison of the resource allocation over time between proportional fair and transaction-based scheduling.

all individuals are evaluated and the best individuals are recombined randomly for the next generation.

For the first generation, the individuals of the GA are initialized randomly with the possibility to have one individual to be the solution of the PF scheduling scheme. This makes it easier to find an initial valid solution which is the starting point for the evolution. After each generation, the crossover operator combines two parent individuals into two offspring individuals by randomly selecting a scheduling decision for either of the siblings. Additionally, two mutation operators exist which either change the resource allocation in a TTI or interchange the allocation of two TTIs of the same individual. These mutation operators are applied randomly to the individuals of the new generation. As a last step, each solution is changed to be a work-conserving allocation. This means that allocations to a transaction that already transmitted all of its data are given to an unfinished transaction.

The results of the GA are presented in Sec. VI and compared against PF scheduling.

VI. Evaluation Results

We compared the GA for a small problem with the optimal solution from the LP. It showed up that the GA achieves up to about 97 % of the sum utility of the optimal solution. As the number of variables grows linearly with transaction number and duration, we can expect a good convergence of the GA for the evaluated scenario.

To evaluate the performance of the proposed context-aware resource allocation, we compare the results of our GA against the PF scheme. For each scheduling scheme, traffic is comprised of the same transaction sizes. For the PF scheduling, we take the heuristic mentioned in Sec. IV, $\bar{r}_n$ initialized with the long-term average of the channel quality and then evolving with an exponential moving average with window size $T_c = 1000$ (PF). To compare the effect of different utility functions between PF and transaction-aware scheduling, we introduce static prioritization in the PF case (denoted by PF_{prio}). This prioritization means that no background traffic is served until all foreground transactions are finished. Finally, we have the transaction-aware (TA) resource allocation which is determined by the GA with 10,000 generations.

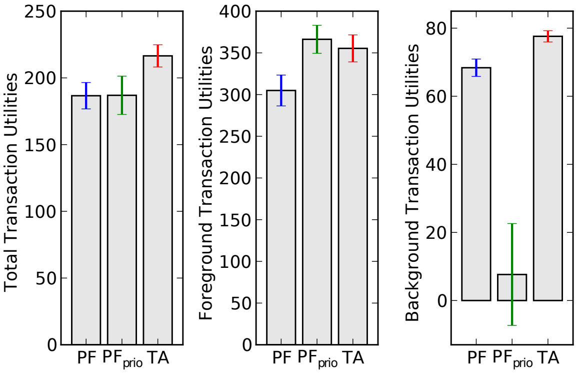

Fig. 4. Average utility comparison between proportional fair (without and with prioritization: PF and PF_{prio}) and transaction aware (TA) scheduling.

The setup is equal to Sec. III. Simulation duration is 10 s with 10 active transactions. The utility functions are assigned in an interleaving manner. This means $T = 0, 2, \ldots$ have an interactive utility function, whereas $T = 1, 3, \ldots$ have the file-download utility function with the slope depending on the transaction size.

Fig. 3 shows an example of the resource allocation with PF and TA. While the PF scheduler targets a high system capacity while maintaining a certain fairness level, the TA scheduling directly tries to minimize the transaction finish times. As a consequence, PF scheduling is mainly guided by channel variations, which is good for the sum rate of the system. For the case of TA scheduling, we can observe that mostly interactive traffic is served first. However, if the GA detects that an interactive transaction has no chance to meet its deadline (as for T_2), it postpones this transaction in favor of the file-download transactions, because it can gain more from the constant slope of these transactions. In a real system, this would mean a drop of the respective service.

In Fig. 4, we show the average utility obtained in 100 optimizations with random traffic. The error bars represent the 95% confidence intervals in all figures. It can be seen that TA scheduling improves the overall average utility in comparison to the PF scheme. In contrast, static prioritization cannot improve this overall transaction utility. This is due to the fact that static prioritization obtains a high utility for the interactive traffic at the cost of file downloads. Large interactive transactions block even the small file-download transactions, so that they suffer from a very bad utility. TA scheduling on the other hand achieves to improve the utility of both traffic types in comparison to PF scheduling.

Fig. 5 demonstrates the utility gains in terms of average transaction finish times. Static prioritization increases the average finish time of PF by a reduction of multi-user diversity from ten users to five users, because maximal five transactions

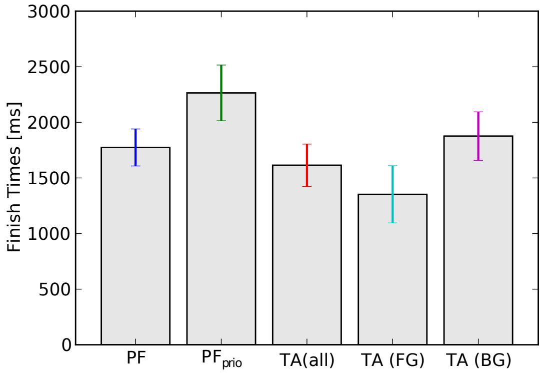

Fig. 5. Comparison of the average transaction finish times for the different scheduling algorithms. Transaction aware scheduling also split up in foreground (FG) and background (BG) traffic.

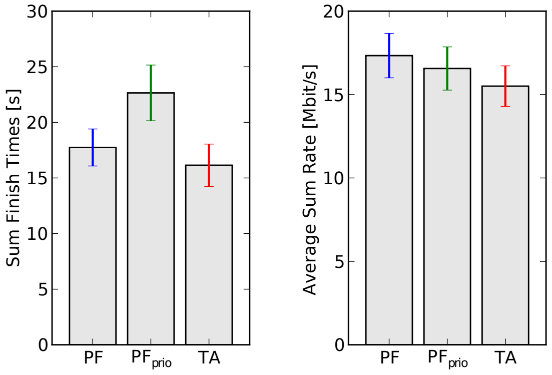

Fig. 6. The left plot compares the sum of finish times of active users, the right plot compares the average sum rate of the evaluated sector for the different scheduling algorithms.

are selectable at any given TTI. TA scheduling achieves slightly smaller average finish times than the PF scheme.

The left plot in Fig. 6 compares the sum of the finish times for the different scheduling schemes. It can be seen that TA achieves to reduce this sum. It tends to keep the transmissions of one transaction together, whereas the PF scheduling fragments the transmissions due to channel variations. This is advantageous for the sum finish time as also illustrated in the schematic of Fig. 2. Again, PF with static prioritization performs worst.

On the right side of the figure, the sum rates are compared. Here, TA performs slightly worse than the PF scheduling schemes because it focuses on finishing transactions early. Consequently, we have less multi-user diversity for the remaining transactions in this traffic scenario.

VII. Conclusion and Future Work

We demonstrated that context-aware resource allocation is worth the effort. While context awareness improves resource allocation, so far it was not clear how to systematically integrate this new approach into scheduling. Our proposed transaction-based framework is one method to do so.

Based on time-variant Utility functions we introduced a new method to minimize the delays for various traffic types. Even by exploiting only the different demands of foreground and background applications, we significantly reduced the transmission delay in a controlled scenario.

However, CARA is neither limited to this simple scenario nor to the studied context feature. Thus, future work needs to:

- Study the gains with further context features such as user environment, UE orientation, or UE processor and memory load.
- Account for the performance degradation due to imperfect context information and signaling costs.
- Design efficient scheduling heuristics that closely approach the above optimal results at lower computational complexity.

Currently, we are developing scheduling heuristics that employ traffic and channel prediction to transform transaction utilities into instantaneous utility functions. We will account for imperfect context information in a next step.

References

[1] W.-H. Kuo and W. Liao, “Utility-based radio resource allocation for QoS traffic in wireless networks,” *IEEE Trans. Wireless Commun.*, vol. 7, no. 7, 2008.

[2] J.-W. Lee and J.-A. Kwon, “Utility-based power allocation for multiclass wireless systems,” *IEEE Trans. Veh. Commun.*, vol. 58, no. 7, 2009.

[3] M. Katoozian, K. Navaie, and H. Yanikomeroglu, “Utility-based adaptive radio resource allocation in OFDM wireless networks with traffic prioritization,” *IEEE Trans. Wireless Commun.*, vol. 8, no. 1, 2009.

[4] F. Kelly, “Charging and rate control for elastic traffic,” *Euro. Trans. Telecomms.*, vol. 8, no. 1, 1997.

[5] P. Viswanath, D. Tse, and R. Laroia, “Opportunistic beamforming using dumb antennas,” *IEEE Trans. Inf. Theory*, vol. 48, no. 6, 2002.

[6] H. J. Kushner and P. Whiting, “Convergence of proportional-fair sharing algorithms under general conditions,” *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, 2004.

[7] R. Irmer (ed.), “Radio access performance evaluation methodology,” NGMN White Paper V1.3, 2008.

[8] “Further advancements for E-UTRA physical layer aspects,” WSG RAN, Tech. Rep. TR 36.814, 2010.

[9] P. Dent, G. Bottomley, and T. Croft, “Jakes fading model revisited,” *Electronics Letters*, vol. 29, no. 13, 1993.

[10] “Selection procedures for the choice of radio transmission technologies of the umts,” ETSI, Tech. Rep. TR101112/UMTS30.03 V3.2, 1998.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[12] J. Nielsen, “Website response times,” http://www.useit.com/alertbox/response-times.html, 2010, link verified on Feb. 3rd, 2011.