Time-aware Utility-based Resource Allocation in Wireless Networks

Calin Curescu, and Simin Nadjm-Tehrani, *Member, IEEE,*

***Abstract*— This article presents a time-aware admission control and resource allocation scheme in wireless networks, in the context of a future generation cellular network. The quality levels (and their respective utility) of different connections are specified using discrete resource-utility (R-U) functions. The scheme uses these R-U functions for allocating and reallocating bandwidth to connections, aiming to maximise the accumulated utility of the system. However, different applications react differently to resource reallocations. Therefore at each allocation time point the following factors are taken into account: the age of the connection, a disconnection (drop) penalty and the sensitiveness to reallocation frequency. The evaluation of our approach shows a superior performance compared to a recent adaptive bandwidth allocation scheme (RBBS). In addition we have studied the overhead that performing a reallocation imposes on the infrastructure. To minimise this overhead, we present an algorithm that efficiently reduces the number of reallocations, while remaining within a given utility bound.**

***Index Terms*— Bandwidth allocation, QoS provisioning, C.2.1.k Wireless networks, Utility-based optimisation, C.2.3.a Network management**

I. Introduction

A key feature of future generation wireless networks is to provide mobile users with multimedia and data services seamlessly. The bursty nature and variable bandwidth needs of most of the new services call for novel treatments of the network resource management so that application needs are satisfied, and at the same time network provider resources are used in the best way. Many existing works in resource allocation focus on one part of this equation to the detriment of the other party. If end-to-end guarantees of user Quality of Service (QoS) requirements are in focus, then some decisions may become counterproductive seen from a system-level perspective, and vice versa. In this article we approach the problem by methods that bridge this gap.

The article presents a bandwidth allocation and admission control mechanism to be used in a radio network cell of a future generation telecommunication network. As the main bottleneck we consider the bandwidth of the wireless link between the user equipment (UE) and the base transceiver station (BTS).

The different nature of the wireless channel (as compared to the wireline) makes the QoS delivery more challenging. First, the fixed capacity of the allocated wireless spectrum makes the system bandwidth-constrained, and hence allocation problems cannot be solved by over-provisioning. Second, the resources available to a user might vary greatly during the lifetime of a connection. Due to mobility, a user might leave a cell where bandwidth is plentiful and enter a congested area. Also, the effective bandwidth of the wireless link may fluctuate due to fading and interferences.

The above three factors describe a system where bandwidth availability is highly variable in time, and the system may often find itself in an overload situation. For the bandwidth manager to take the best allocation decisions, we assume that a quantitative measure of the utility (benefit) generated by each connection is available. One way to capture the application-specific perceived quality depending on resource availability is via resource-utility (R-U) functions.

Consequently, a straightforward allocation optimisation criterion (that can be easily linked to network operator revenues) is maximising the system utility. This can be calculated as the sum of the utility of each connection[1].

Moreover, for such an open dynamic system, resource reallocation might be needed in order to improve total utility (if bandwidth becomes available, i.e. a connection finishes or leaves the cell) or to provide graceful degradation (when bandwidth has to be reallocated to new connections or incoming handovers). In order to make more informed decisions on resource reallocation, in addition to utility functions, we also consider the fact that different applications react differently to resource reallocation. For example, if a hard real-time application is degraded, we would expect no utility from this application, and the resources invested so far would be wasted. On the other hand, an FTP session will have no restriction to switch between different resource allocation levels, no matter how often.

Therefore, we propose a Time-Aware Resource Allocation scheme (TARA) that aims to provide bandwidth allocation/reallocation based on the utility-efficiency (utility per bandwidth) of the competing connections. The novelty is that our scheme identifies how resource reallocation decisions affect the utility of the application, and integrates this information into the bandwidth management algorithm. Based on their flexibility to reallocations, we have categorised applications in three classes: non-flexible, semi-flexible and fully flexible. The time at which a reallocation decision is taken is also very important. Because of the invested resources, disconnecting a

The authors are with the Department of Computer Science, Linköping University, Linköping, Sweden. Email: [calcu,simin]@ida.liu.se

[1]Since each connection represents an application the two terms will be regarded as interchangeable in the following text.

connection when it is nearly finished creates a bigger utility loss than if it is dropped just after start and bandwidth has been invested for a small period of time. Thus, the system has to be aware of the age of the connections to take a good (re)allocation decision. In addition to this, two more factors have been considered when reallocating. First, the dropping penalty allows the user to specify its dissatisfaction of being first accepted and then rejected. Second, the sensitivity of some of the connections with respect to the frequency of reallocations is considered.

To evaluate our scheme we have built a simulation platform in which we compare our approach with a base-line version that is unaware of the previously mentioned factors and a recent published algorithm, namely the Rate-based Bandwidth Borrowing Scheme (RBBS) [7].

Finally we consider the overheads created by our bandwidth allocation scheme as a result of the periodic reallocation. This increases the resource demand from the infrastructure (e.g. CPU time for executing associated control functions or additional bandwidth for signalling). Thus, by performing too many reallocations in order to improve utility, the system might get overloaded. Consequently, service availability will suffer, and the generated utility will decrease; contrary to what was intended. We present and evaluate a new algorithm for controlling this overhead.

The article is organised as follows. In Section II we review other approaches to QoS provisioning. Section III presents background information about resource-dependent utility maximisation. In Section IV we identify the factors affecting the utility of a connection if reallocations are performed and in Section V we show how we include them in our scheme. Section VI describes the evaluation setup and Section VII presents the simulation results of our allocation scheme. In Section VIII we present the above-mentioned overhead considerations. The resulting conclusions are presented in Section IX.

II. Related work

Research on QoS provisioning may pursue different goals. While some research is geared towards end-to-end architectures [2], others address issues at end-system level or network layers. Mechanisms like Intserv [5] and RSVP [6] or Diffserv [4] provide the means of enforcing the necessary QoS parameters (like bandwidth, delay, packet loss probability).

Many applications can be run at different QoS levels, corresponding to a range of resource allocations. These can be used for relative differentiation between applications, but without a notion of “importance”, the QoS management system will not be able to prioritise allocations during overloads. For meeting these needs, utility functions provide an appropriate way to specify a quantitative measure of the QoS perceived by the application [11], [12]. A utility function is similar to a “QoS contract negotiation”. The user specifies all its options and the provider chooses one of them. The advantage of utility functions over run-time negotiations is that the management system knows a-priori about the value corresponding to different resource allocations and might be able to enforce an optimised solution. Chen Lee et al. [11] use resource-utility functions in a QoS management framework with the goal to maximise the total utility of the system. They propose two approximation algorithms, and compare the runtimes and solution quality with an optimal solution based on dynamic programming. In our work we build on top of such an utility maximisation algorithm, but we also take into account bandwidth reallocations and their effect on the connections’ generated utility.

Another important factor for our scheme is the dynamic nature of the environment. Some other approaches [15], [7], are geared towards mobile networks and proposes adaptive bandwidth allocation schemes without an explicit use of utilities. These use a flexible allocation approach, where connections specify a mandatory minimal bandwidth and an ideal maximal bandwidth. Also, both schemes differentiate between real-time and best-effort connections. In the work of Oliveira et al. [15], the allocated amount of bandwidth during the stay in a cell is fixed, it can be changed only at a handoff. El-Kadi et al. [7] provide a more adaptive scheme, by allowing fixed portions of bandwidth to be borrowed from already accepted connections. Although the scheme is adaptive, it does not include a quantitative measure of the importance of the different connections.

Next we present several systems that allocate resources based on a certain “maximisation technique” of utility functions, and also work in highly dynamic environments. Resource assurance (more or less), is a concern for all of these systems.

Rui-Feng Liao et al. [12] use “utility functions” in a bandwidth allocation scheme for wireless packet networks. However as opposed to maximising the total utility of the system, they provide “utility fair allocation” to the connections. Their algorithm extends “max-min fair allocation”, with utility replacing bandwidth as the fairness criterion. While this scheme provides equality to all connections, it might have counterproductive effects during overload conditions, since it degrades all the existing connection to a low common utility.

Abdelzaher et al. [1] propose a QoS-adaptive Resource Management System for Internet servers. A QoS contract is used to specify acceptable QoS levels, along with their utility. There is no restriction in reallocations (similar to our fully-flexible class). However, there is a “minimum” allocation level that must be guaranteed. Otherwise a “QoS violation” penalty (similar to our drop penalty) is incurred. They compare an optimal allocation policy based on dynamic programming with a first-come first-serve policy where resources are not reallocated.

A system that also addresses resource allocation in mobile networks is the “TIMELY Architecture” proposed by Bhargavan et al. [3]. While we are concerned with the allocation at a “policy level” their system coordinates allocation from the MAC-layer, through resource reservation and resource adaptation to the transport layer. Moreover, an end-to-end allocation over both wireline and wireless links is attempted. They employ a revenue model with a 4-tuple: revenue function, termination credit (similar to our drop penalty), adaptation credit (similar in function to what we will call adaptation time) and an admission fee. Maximising the revenue (based

on the max-min criterion) is one of the criteria used during allocation and adaptation. On the other hand, the same 4-tuple is used for all flows. While simplifying allocation, it prevents differentiation (as different importance or different assurance needs) between flows. In our work we assume that the QoS specification (as R-U functions, flexibility classes) is connection specific, and during allocation the system uses these parameters to differentiate between connections.

An optimal sampling frequency assignment for real-time wireless sensor networks is proposed by Liu et al. [13]. In their model, the underlying network uses dedicated channels to communicate between neighbours such that interferences are avoided. Nevertheless, a flow traverses several channels so the bandwidth allocation of the different wireless channels is not independent (as opposed to our work). Utility loss index (that depends on the sampling frequency and is of convex form) characterises QoS and must be minimised across the network in order to optimise the system. Two algorithms are proposed, a centralised that is better suited to small networks and a distributed one that converges in several iterations, and is better with large networks.

In a more classical real-time approach, in their QoS provisioning system [16] Richardson et al. take a lower layer approach, by using value based and real-time scheduling techniques and working at the packet scheduling level. The priority of each packet depends on the value of the connection it belongs to and on its deadline. Total system utility is used to measure system performance.

III. Background

To explain how our bandwidth allocation scheme works, we must first present the notion of bandwidth dependent utility function, and a utility maximisation algorithm.

A. Application utility

The utility of an application (and its associated connection) represents the value assigned by the user to the quality of the application's results. In order to evaluate the utility generated by different resource allocations, we assume that each connection has a resource-utility (R-U) function, which is specified by the user of this connection, $u_i : \mathbb{R}^* \rightarrow \mathbb{R}^*$, i identifies the connection and $\mathbb{R}^*$ is the set of non-negative rational numbers, and $u_i(r)$ describes the utility that accrues given a resource level r. In this paper the resource is the wireless bandwidth in a cell. As a reflection of variety of applications, utility functions may exhibit different patterns: concave, convex, linear or step functions, the only restriction being that a R-U function should be non-decreasing.

For the ease of representation, and to keep complexity low, it is necessary to quantise the utility function using a small set of parameters. Thus, the utility functions is represented by a list of bandwidth-utility pairs, in increasing order of resource consumption [11]:

$$u_i = \left\{ \binom{U_{i,1}}{B_{i,1}}, ..., \binom{U_{i,k}}{B_{i,k}} \right\}$$

where k is the number of utility levels of connection i and $U_{i,1}$ represents the resulting utility if $B_{i,k}$ is allocated to the connection.

B. System utility maximisation

Next we describe a general resource allocation problem. We assume that the utility of a system is the sum of the utilities of all applications in the system. Then the utility maximisation problem can be formulated as follows:

$$maximise\ u(b_1, ..., b_n) = \sum_{i=1}^{n} u_i(b_i)$$

$$subject\,to \sum_{i=1}^{n} b_i \leq B\_max$$

where $u : \mathbb{R}^{*\,n} \rightarrow \mathbb{R}^*$ is the system-wide utility, b_i are the allocation variables to be solved, and B_max is the total available resource.

The above allocation optimisation problem is an NP-hard problem closely related to the knapsack problem; Lee et al. present several approximation algorithms to solve it. As a basic ingredient in our scheme we use one of the algorithms proposed by Lee et al. that we further refer to as $convex\_hull\_opt$ (referred as $asrmd1$ in [11]). Despite its low complexity, the algorithm generates solutions close to the optimal solution [10], [11]. As a first step it first approximates all R-U functions by their convex hull frontier, which are piece-wise linear, concave functions. Next, all convex hulls are split in segments corresponding to their linear parts. Note that a "segment bandwidth" is its projection on the x-axis (the bandwidth increment between two levels). Then all the segments are ordered by a decreasing slope order (see Figure 1), and bandwidth is allocated in this order until depleted. Thus, a connection has allocated an amount equal to the sum of the bandwidths of its "allocated segments". The concave form of the convex hull ensures a consistent allocation. Note that the slope of each segment, $(U_{i,j} - U_{i,j-1})/(B_{i,j} - B_{i,j-1})$, represents its efficiency in terms of contribution to the system utility.

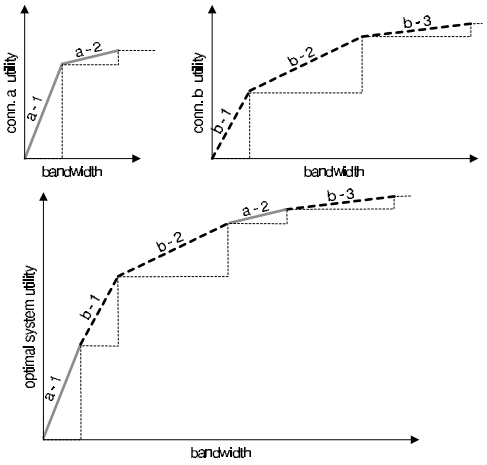

Fig. 1. Optimal allocation order

IV. Reallocation consequences

The R-U functions present the bandwidth-utility dependency in a static manner. In a dynamic system, where resources need to be reallocated, the utility given by a R-U function will represent only a momentary value ($u_i(t)$). A better measure of the utility generated by a connection would be its accumulated utility in time, which is the utility generated by the connection over its entire duration.

If, for some application class, the accumulated utility of a connection (u_i^a) corresponds to the integral of all the momentary utilities, that is $u_i^a = \int_0^T u_i(t)dt$, then the following equality holds:

$$u^a = \sum_{i=0}^{n} u_i^a = \sum_{i=0}^{n} \int_0^T u_i(t)dt = \int_0^T \sum_{i=0}^{n} u_i(t)dt = \int_0^T u(t)dt$$

where u^a denotes the system-wide utility accumulated over time and $u(t)$ is the momentary system-wide utility. T represents a time interval. The above equations show that in this case, the maximisation of u_i^a can be achieved by maximising $u_i(t)$ at each time point t.

However, for other application classes $u_i^a \neq \int_0^T u_i(t)dt$. For example, there are applications with strong needs for resource assurance. That is, if the initial agreed resource amount is degraded, then all the potential utility generated until that moment is lost. In the end $u_i^a = 0$, and resources allocated to it since its arrival have been wasted. Therefore, our allocation algorithm needs to take into account the effect that reallocations have on the accumulated utility of the connections.

The main advantage of using utility functions is that the user (application) has the possibility to quantitatively specify the value he is attaching to the results. Accepting a connection and allocating a certain amount of bandwidth is similar to negotiating and signing a a QoS contract between the user and provider. Since the R-U functions provide a framework to specify all the acceptable levels, the negotiation phase (and the associated overheads) can be skipped. In the same line of thought, each reallocation would amount to a breach of contract and signing of a new contract. Because of different application types or user preferences, different connections have different tolerance to bandwidth reallocation. The challenge is to find a representative (and small) set of parameters that satisfactorily describe the reallocation effects for a large set of applications. The vision is to give the user the possibility to specify these parameters so that the system can take the right (re)allocation decision. Similarly to the R-U functions they convey user expectation and should not be regarded as a-priori fixed parameters tuned by the system operator.

We have thus identified three factors that affect the perceived accumulated utility of a connection: the flexibility (adaptability) to reallocations, the sensitivity to a complete disconnection (as opposed to only a bandwidth reduction), and the sensitivity to the frequency of reallocations.

A. Flexibility classes

We first divide the applications into three broad classes depending on their flexibility with respect to reallocations.

Class I represents non-flexible connections. They require strict resource assurance to fulfill their mission. That is, once accepted (with initial utility u_i^{init}), the resource amount cannot be re-negotiated. If the management system cannot assure the initial resource amount at any time-point during the lifetime of the connection, there will be no utility gained for the whole duration of the connection, and already invested resources are wasted. If accepted, any subsequent reduction of bandwidth is equivalent to dropping the connection. Since it uses the same amount of resources during its lifetime, increasing the bandwidth brings no benefit. If the connection is not dropped, the accumulated utility of the connection is calculated by this formula: $\boldsymbol{u_i^a = u_i^{init} \times duration}$. Examples are hard real-time applications, critical control data, real-time data streams.

Class II represents semi-flexible connections. These are applications that are judged by their worst moment in their lifetime. For this type of connection the lowest utility (respectively bandwidth) experienced during its lifetime is used for calculating the utility for the whole duration: $\boldsymbol{u_i^a = u_i^{min} \times duration}$. Compared to class I, a resource degradation, while diminishing utility, is not disastrous. However, once a certain level reached, the results cannot be improved if the resource allocation is increased at a later point. For example, users often remember the worst portion of a multimedia stream, or a distributed game. Another good example is sensor readings where the resolution bound is important. The resolution of the whole stream is the lowest resolution from all the readings.

Class III represents fully-flexible connections. These are the connections with no real-time requirements, and they can adapt to both increases and decreases of the bandwidth. The accumulated utility is the sum of all the momentary utilities over the total duration: $\boldsymbol{u_i^a = \int_0^{duration} u_i(t)\, dt}$. Examples are fetching e-mail, file transfer, or any type of connection in the "best effort" category.

A real-word application could be a combination of different connections of different class. For example it could consist of two parts, a mandatory one that is class I and a fully flexible, class III. For this paper we consider applications to belong to only one of the above classes. Note that the shape of the R-U function does not depend at all on the class of the connection. The class does not affect the initial allocation possibilities, but only describes the effects at subsequent reallocations.

B. Drop penalty

We assume that disconnecting (dropping) a connection before its natural end will bring its accumulated utility to zero. This shows that invested resources will be wasted by such a decision. In a similar manner resources have been invested on the user side, and will be lost. Therefore, the user should be able to specify a certain drop penalty, which represents the customer dissatisfaction when a connection is disconnected after being admitted into the system. Let P_i^{drop} be the penalty for dropping a certain ongoing connection. If disconnected, the final utility of the connection $u_i^a = -P_i^{drop}$. If utility is used in calculating the revenue of the network operator, a negative utility will imply some form of compensation to the user.

C. Adaptation time

Flexible (class III) applications can adapt to both increases and decreases in bandwidth. In a dynamic environment, these connections might be subjected to very frequent reallocations. But even these flexible applications might need a certain amount of time to adapt to the new "mode" after a reallocation. For example, some algorithms for encoding, encryption, compression could be changed, some computations need to be restarted. Performing frequent reallocations might be worse than keeping a connection at a constant, lower resource level. A specified adaptation time is a way to reflect the minimum time between reallocations in order to keep the expected utility.

The effects of not respecting a "minimum adaptation time" could greatly differ for different connections. Nevertheless, we propose the following performance degradation model. If the time between two bandwidth reallocations (I_i) is less then a specified adaptation time (A_i) then we assume that the utility generated during this interval is only I_i/A_i of the utility under normal circumstances, thus characterising a penalty for frequent reallocations. Classes I and II should not be subject to frequent reallocations (bandwidth increases are useless, decreases are few and bound by the allocation levels of the utility function). Thus, this penalty is meaningful only for class III connections.

V. Dynamic reallocation

Because of the highly dynamic environment, constant reallocation is needed in order to obtain the best results. Basically whenever a new connection or handover request arrives or a connection ends, a new reallocation might be needed to improve system utility.

In a large system, this event-based allocation may lead to an unacceptable high call rate to the (re)allocation algorithm. This overhead can be controlled by employing a periodic (re)allocation method. New connections and handovers are put in a queue and will be processed at the beginning of the next allocation period. For reasonably low values of reallocation period, the process will be transparent to the user.

Note that ongoing connections are treated at the same time as requests for new connections or handovers. For the latter two the (re)allocation algorithm plays also the role of admission control. Only requests that are allocated a bandwidth greater than zero are admitted.

In Section III-B we mentioned an near-optimal allocation algorithm that uses the R-U functions as input. The algorithm will order and accept connections based on their efficiency. We keep this as a base allocation algorithm. However, since the original R-U functions do not describe the history of a connection, we have to take the additional factors that describe the effects of reallocations (see Section IV) into account. Therefore we create "artificial" R-U functions by modifying the original R-U functions at every (re)allocation time point. For instance, in an ongoing class I connection, resources have been invested for some time. The corresponding potential utility however, will only be gained if the connection is not dropped. When compared to a new connection, the ongoing connection comes with this so far earned utility, that effectively increases the efficiency of the connection over the rest of its lifetime. Thus by modifying the R-U functions we make the connections of all ages and classes comparable efficiency-wise.

While at each allocation point we optimise based on the utility-efficiency of the different connections, only a clairvoyant algorithm that knows all the future arrivals could provide a truly optimal allocation. For instance a class I connection accepted at some point might be dropped if an increased number of higher efficiency connections arrive at a later point. By not accepting it in the first place the system would have avoided paying the drop penalty. As clairvoyance is not a realistic assumption the only other possibility would be do predictions based on profiling past arrivals. Even this option is considered as unrealistic in our context.

In the next subsections we will explain how the R-U functions are modified at each reallocation. Table I summarises the parameters utilised in the process.

TABLE I
NOTATION SUMMARY

u_i	The original R-U function of conn. i
u_i^{age}	The age-modified R-U function of conn. i
u_i^{drop}	The drop penalty modified R-U function of conn. i
u_i^{adapt}	The adaptation-time modified R-U function of conn. i
$b_i(l)$	The bandwidth of conn. i corresponding to QoS level l
p_i^{drop}	The drop penalty of conn. i
t_i^{max}	The duration of conn. i
t_i^{age}	The current age of conn. i
I_i	The time passed since the last allocation for conn. i
A_i	The adaptation time of conn. i
u_i^a	The value of the time-accumulated utility for conn. i
u^a	The value of the system-wide time-accumulated utility

A. Age and class dependent modifications

To get a feeling for why age modifications are needed, we start by giving an example of a reallocation decision where the original, unmodified R-U function is used. Assume there is a class I or II connection $conn_1$, which has an R-U function that evaluates to 3 for bandwidth 4 ($u_1(4) = 3$). Assume the total duration of the connection $t_1^{max} = 10$ seconds of which 5 seconds have elapsed, denoted by t_i^{age}, and the allocated bandwidth during this time was $b_1 = 4$. This means that the accumulated utility so far $u_1^{curr} = u_1(b_1) \times t_1^{age} = 3 \times 5 = 15$. At this time a new connection $conn_2$ is competing with the old one for the same bandwidth. Assume that for $conn_2$ the utility corresponding to bandwidth 4 is 5 ($u_2(4) = 5$). Because the $convex\_hull\_opt$ allocation algorithm is using the slopes (utility/bandwidth) of the R-U functions' convex hulls to make decisions, and $3/4 < 5/4$, it will choose $conn_2$ in comparison with $conn_1$ and u_1^{curr} will be lost. Let's see what is the utility gained by the system after the next 5 seconds: $u^a = u_2(4) \times 5 = 5 \times 5 = 25$. If the first connection had been kept, the utility would have been $u^a = u_1(b_1) \times t_1^{max} = 3 \times 10 = 30$, thus the swapping decision is wrong. Therefore, to replace an old connection with a new one, the utility generated by the new connection until the completion time of the old connection should be greater than the utility generated by the old connection during its entire life time (see shaded areas in Figure 2). In our example, $conn_1$ should be swapped with $conn_2$ only if $u_2(4) \times 5 > u_1(4) \times 10$.

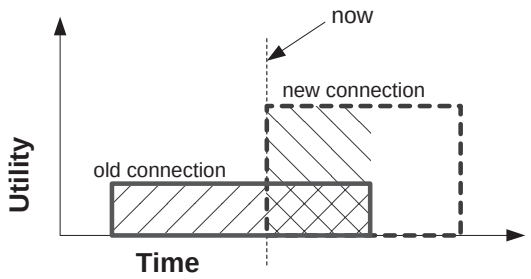

Fig. 2. Replacement opportunity

In the above example we assumed that we have the choice only to swap $conn_1$ with $conn_2$, at the same bandwidth level. In general, we need to consider a connection that has multiple acceptable bandwidth levels (in its R-U function), of which one is the current allocation. To reflect the age, and thus the accumulated utility at the reallocation time point, we need to modify the R-U function of the existing connection as follows. Each allocation corresponds to a level l in the R-U function, where $1 \leq l \leq k$, and k is the maximum number of levels. We denote the bandwidth at level l by $b_i(l)$, and the corresponding utility by $u_i(b_i(l))$. For instance, in Figure 3 (a): $b_i(1) = 0$, $u_i(b_i(1)) = 0$, $b_i(2) = 2$, $u_i(b_i(2)) = 1$, etc. Let the already existing allocation level at a reallocation time point be j. In Figure 3, $j = 3$ and $b_i(j) = 4$.

Fig. 3. Age modification for class I and II, with $\boldsymbol{t_i^{age} = 5}$, $\boldsymbol{t_i^{max} = 10}$, and actual bandwidth $\boldsymbol{b_i = 4}$

Here we explain how a modified R-U function for a class II connection is constructed and refer to Figure 3 (a) and (b) as an example. When constructing the modified R-U function, we can divide the allocation levels in two sets (corresponding to two situations). If the bandwidth allocation stays the same or is increased, u_i^a remains the same. Thus, for all these levels, the modified R-U function, denoted by u_i^{age}, will be equal to the utility for the existing allocation:

$$u_i^{age}(b_i(l)) = u_i(b_i(j)) \qquad \forall j \leq l \leq k$$

Decreasing the bandwidth results in losing a portion (or all) of the connection's accumulated utility so far. So we first compute the lost utility:

$$u_i^{lost}(l) = \Big(u_i(b_i(j)) - u_i(b_i(l))\Big) \times t_i^{age} \quad \forall 1 \leq l < j$$

Then we reduce the utility that can be accrued at the respective levels by modifying u_i to u_i^{age}:

$$u_i^{age}(b_i(l)) = u_i(b_i(l)) - \frac{u_i^{lost}(l)}{t_i^{max} - t_i^{age}} \qquad \forall 1 \leq l < j$$

Note that for $l < j$ there is a larger difference between two adjacent utility levels in u_i^{age} compared to u_i; that is, the slopes of the segments of the convex hull of u_i^{age} (for $l < j$) are steeper. That is, a reduction of the allocation can only be compensated by a higher utility gained from other connections. More precisely, the slope increase in the modified R-U function is exactly large enough so that, if bandwidth is reallocated to other connections, the newly accepted (improved) connections will generate not only a higher utility for the reallocated bandwidth, but in addition also recover the utility lost by degrading this connection. The lost utility will be recovered during the interval $t_i^{max} - t_i^{age}$ (that is, before the time point at which the degraded connection would have released the bandwidth naturally).

For class I connections, any decrease in bandwidth means the connection is dropped (leads automatically to 0 bandwidth). The modified R-U function for a class I connection is presented in Figure 3 (c), if the original R-U function was the same as that depicted in Figure 3 (a). For this class, the zero-bandwidth level is calculated similarly to the class II case. Class III connections do not lose utility (waste resource) in case of a reallocation, thus their original R-U function needs no age modification.

Now the question becomes, do we assume that the real duration of every connection is known? Obviously this is too unrealistic to assume. In practice we have to resort to an estimate of a connection's duration. The better the estimation of the connection duration, the more accurate the modification will be. This is because overestimating/underestimating the duration of a connection will underestimate/overestimate the importance of a bandwidth decrease for this connection. In Section VII-A we further discuss how the system behaves in the absence of an exact knowledge of the duration.

B. Drop penalty influence

Class I and II connections are dropped (disconnected) whenever their momentary bandwidth becomes zero, since that connection yields no utility in the end. Class III connections should not be dropped because of bandwidth shortage, since they can recover at a later time, without penalty. Recall that each connection comes with its own drop penalty, P_i^{drop}. To reflect this sensitivity to disconnections, the R-U function is further modified (the effect is additional to the age-dependent modification) as follows:

$$u_i^{drop}(b_i(l)) = \begin{cases} u_i^{age}(b_i(l)) - \dfrac{P_i^{drop}}{t_i^{max} - t_i^{age}} & \text{for } l = 1 \\ u_i^{age}(b_i(l)) & \forall l \neq 1 \end{cases}$$

Similar to u_i^{lost} in the previous subsection, in order to improve the accumulated utility, this penalty should be recovered before the natural end of the connection.

Note that the modification is only applied to the first level (where $b_i(1) = 0$), because if bandwidth is not reduced to zero, the connection is not dropped. Figure 4 presents the further modification of the class II R-U function from Figure 3 (b) given a drop penalty $P_i^{drop} = 8$.

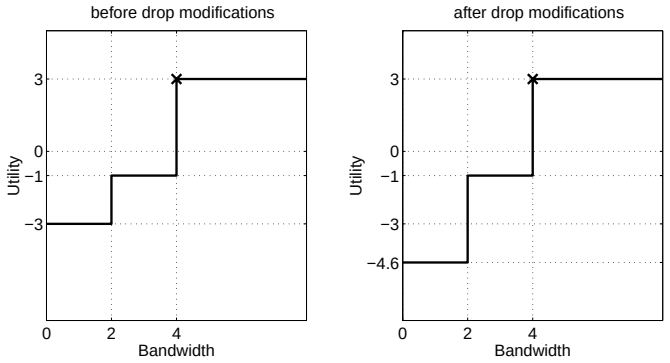

Fig. 4. Class II drop modification with $\boldsymbol{P_i^{drop} = 8}$, $\boldsymbol{t_i^{age} = 5}$, $\boldsymbol{t_i^{max} = 10}$, and actual bandwidth $\boldsymbol{b_i = 4}$

C. Adaptation time influence

This modification, that reflects sensitivity to bandwidth fluctuations, is only applied to class III connections. Classes I and II will not be subject to frequent reallocations (bandwidth increases are useless, decreases are bounded by the number of the R-U function levels).

As presented in Section IV-C, for a flexible class III connection, if reallocation is performed before the adaptation time required by the application ($I_i < A_i$), the gained utility in this interval (normally $u_i \times I_i$) is diminished to an I_i/A_i of the normal gained utility. This could be seen as a new form of penalty computed as $P_i^{adapt} = u_i \times I_i \times (A_i - I_i)/A_i$, to be subtracted from u_i^a.

Each time there is a reallocation, the R-U function is modified to represent the sensitivity to the current reallocation frequency. If $I_i < A_i$ and there is a change from the current allocation level (j) then an adaptation penalty is incurred:

$$u_i^{adapt}(b_i(l)) = \begin{cases} u_i(b_i(l)) - \dfrac{P_i^{adapt}}{I_i} & \forall l \neq j \text{ and } I_i < A_i \\ u_i(b_i(l)) & \text{for } l = j \text{ or } I_i \geq A_i \end{cases}$$

An example of modifications depending on adaptation time is shown in Figure 5.

D. Algorithm overview

To summarise, Figure 6 presents a high-level version of our allocation algorithm, that is invoked periodically and independently for each cell of the network. The methods containing "modify" in their name construct the modified R-U functions as described in the previous sections. Some of the parameters used in the algorithm are presented in Table I, and the following are added: $class_i$ is the connection class, b_i is the current allocated bandwidth, new_b_i the new allocation decision, and b_i^{min} the lowest bandwidth granted in the connections' lifetime, and $period$ represents the running

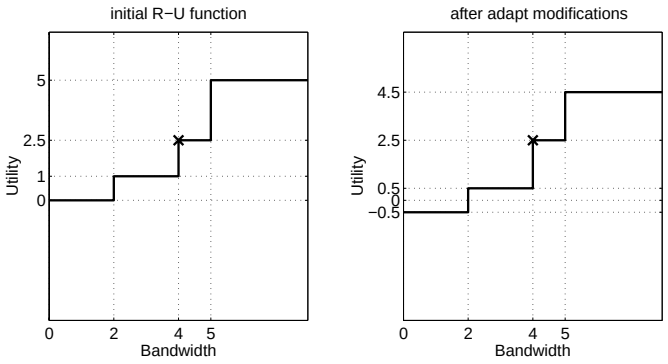

Fig. 5. Class III adaptation modification with $\boldsymbol{A_i = 5}$, $\boldsymbol{I_i = 4}$, $\boldsymbol{P_i^{adapt} = 2}$, and actual bandwidth $\boldsymbol{b_i = 4}$

periodicity of the algorithm. As input the algorithm has all n connections that want bandwidth in this cell (new, old and handed over). Besides the proper (re)allocation algorithm we present also the utility accounting algorithm, that once an allocation decided, calculates the up-to-date system-wide utility, that is our main performance metric.

Bandwidth (Re)allocation & Utility accounting Algorithm:
input: $\forall 1 \leq i \leq n$: $b_i, u_i, class_i, t_i^{max}, t_i^{age}, P_i^{drop}, A_i, I_i, b_i^{min}$
output: $\forall 1 \leq i \leq n$: new_b_i //result of this allocation
$\forall 1 \leq i \leq n$: u_i^a, u^a //result of utility accounting

Bandwidth (Re)allocation:
for $i := 1$ **to** n **do** //modify the R-U functions
  if $class_i = I$ or $class_i = II$ **then**
    $u_i' :=$ age_modify(u_i, $class_i$, b_i, t_i^{max}, t_i^{age});
    $u_i' :=$ drop_modify(u_i', $class_i$, P_i^{drop}, t_i^{max}, t_i^{age});
  if $class_i = III$ **then**
    $u_i' :=$ adapt_modify(u_i, $class_i$, b_i, A_i, I_i);
$(new\_b_1, ..., new\_b_n) :=$ convex_hull_opt($u_1', ..., u_n'$) //new allocation computed

Utility Accounting:
for $i := 1$ to n **do**
  if $class_i = I$ or $class_i = II$ **then**
    if ($class_i = I$ and $new\_b_i \leq b_i$) or ($class_i = II$ and $new\_b_i = 0$) **then**
      $u_i^a := -P_i^{drop}$; //rejected, apply drop penalty
    else //not rejected
      $u_i^a := u_i(b_i^{min}) \times t_i^{age}$; //set accum. utility so far
  if $class_i = III$ **then** //update accum. utility so far
    $u_i^a := u_i^a + u_i(new\_b_i) \times period$;
    if $I_i < A_i$ **then** //apply adaptation penalty
      $u_i^a := u_i^a - u_i(b_i) \times I_i \times (A_i - I_i)/A_i$;
    if $new\_b_i \neq b_i$ **then** //mark new reallocation
      $I_i := 0$;
    else $I_i = I_i + period$;
$u^a := \sum_{i:=1}^{n} u_i^a$;

Fig. 6. Algorithm overview

VI. Evaluation setup

To evaluate the advantage of using utility-based characteristics of a connection we have compared our scheme, the "Time-aware resource allocation scheme" (TARA) , with a recent flexible allocation scheme that addresses similar network problems. We begin with a short description of the "Rate Based

Borrowing Scheme" (RBBS) proposed by El-Kadi et al. [7]. We then explain how we have reconstructed that algorithm in our simulation environment to make valid comparisons (by ensuring that the choices of parameters were compatible and reproducing their earlier results).

The RBBS paper proposes an admission control and bandwidth allocation scheme for cellular networks. In order to not deny service to requesting connections (both new or handovers), bandwidth will be borrowed from already accepted connections. The algorithm uses the following strategy. Each connection that arrives in the system comes with a minimum (min_i) and a maximum (max_i) bandwidth requirement. The actual borrowable bandwidth (abb_i) is calculated as a fraction (f) of the difference between maximum and minimum bandwidth, $abb_i = f \times (max_i - min_i)$, where f is a cell-wide parameter. Another cell-wide parameter is λ which is the number of equal shares the abb_i is divided into. When there is not enough bandwidth available at a certain admission point, bandwidth is freed by decreasing the allocation to all connections with one level (a share from the abb_i). Moreover, in order to provide a smooth change in bandwidth allocation, only one share from the borrowable part can be lent at any time. RBBS divides connections in two classes. Class I are considered real-time connections and a certain amount (e.g. 5%) of the cell bandwidth is reserved to be exclusively used for handovers for this class. This is because class I connections should have always (and can be handed over with) at least min_i bandwidth allocated, otherwise they should be dropped. Class II applications are considered best-effort and can be handed over with any allocated bandwidth (greater than zero) in the new cell. Requests for new connections (both class I and II) are treated more strictly, they are only accepted if enough bandwidth is available to accommodate them at the same level as the cell. When connections terminate or are handed over the available bandwidth increases. If there are connections degraded below the cell level (due to handovers), they will be upgraded first. Otherwise, when enough bandwidth becomes available, the whole cell moves to a better QoS level.

The above work is very interesting because the authors take into consideration many of the characteristics of a Third Generation (3G) network. They consider different traffic types (described next), with different bandwidth requirements, multiple allocation levels, resource assurance classes, etc. On the other hand they do not use a quantitative performance metric (such as utility), that could glue such a complex system together and steer allocation, but use the usual performance metrics such as blocking/dropping probabilities, that are more suited for fixed allocation / single service systems (e.g. 2G). We will return to this issue later in the paper.

To get a good comparison of our scheme and the RBBS we have used the same traffic characteristics as those used for evaluation of RBBS [7]. The same traffic mix has been used first by Oliveira et al. [15] as representative for future mobile communication networks. The first 9 columns of Table II are identical with the ones in the RBBS paper. As in their experiments, the requested bandwidth and connection duration are not fixed, but follow a geometric distribution with the given minimum, maximum and mean values (columns 2 to 6)

TABLE II

TRAFFIC MIX USED IN THE EXPERIMENTS

Applic. Group	Bandwidth Requirement (Kbps)			Connection Duration (sec)			Examples	RBBS class	TARA class	Relative utility per bit
	min	max	avg	min	max	avg				
1	30	30	30	60	600	180	Voice Service & Audio Phone	I	I	1
2	256	256	256	60	1800	300	Video -phone & Video -conference	I	II	1/3
3	1000	6000	3000	300	18000	600	Interact. Multimedia & Video on Demand	I	II	1/10
4	5	20	10	10	120	30	E-Mail, Paging, & Fax	II	III	3
5	64	512	256	30	36000	180	Remote Login & Data on Demand	II	III	1/5
6	1000	10000	5000	30	1200	120	File Transfer & Retrieval Service	II	III	1/7

The minimum acceptable bandwidth is fixed and presented in Column 2. The second column from right represents how we mapped the different application groups into to our connection classes (non-adaptive, semi-adaptive, fully-adaptive).

Since the RBBS is not based on utilities, we had to associate each of the 6 application groups with an R-U function shape. For example, the shape of the R-U function for application group 3 (the one representing interactive multimedia) is presented in Figure 7. All R-U functions that we used, follow the minimum and maximum bandwidth requirements as specified in Table II.

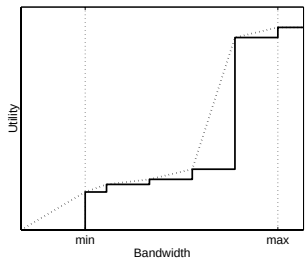

Fig. 7. R-U function shape for group 3

The rightmost column in Table II reflects a relative importance between application groups. For example, since one might be ready to pay roughly three times more for a video-phone conversation, which has a bandwidth demand of 256 Kbps, the utility per bit should be almost three times higher for an audio-phone that requests only 30 Kbps. It represents the utility per bit associated with the requested bandwidth, e.g. if the requested bandwidth of a connection in application group 3 is $6{,}000$ Kbps then the utility for this bandwidth is $6{,}000{,}000 \times 1/10 = 600{,}000$. Having fixed the relative difference at the maximum level, all the other utility values of the R-U function are calculated following the shape of the function. While assigning utility values is always a subjective problem, we tried to use early practice values in our experiments. Ruben et al. [17] performed a study at Ericsson Cyberlab in Singapore and had access to current conceivable business models.

In our simulation environment, connections arrive at the user equipments (UE) following an exponentially distributed inter-arrival time with a mean of 15 minutes. All the 6 application groups arrive with equal probability. Mobility is modelled in the following way: the time at which a UE changes cell (and requests a handover if a connection is ongoing) follows a geometric distribution starting from 60 sec and mean 300 sec,

with equal probability to move in any of the neighbouring cells. Fluctuations of the wireless link, mentioned as a source of bandwidth variability in the introduction, have not been implemented in the simulator. Nevertheless, the random handover and new connection arrival together with the different sizes and R-U functions of the connections ensure a very dynamic resource variability. We believe that since our system deals with this variability properly, the radio link variability can be dealt with analogously.

Our simulations were performed in a simulation environment described by Jonasson [9] and built on top of JavaSim, a component-based, simulation environment developed at Ohio State University [18], [8]. We have simulated a hexagon cell-grid of 16 cells, 4×4 , and a go-around world model to preserve uniformity in our grid. Each cell has a capacity of 30 Mbps.

For all the schemes the bandwidth allocation/reallocation has been performed with a period of 2 seconds. The drop penalty was set using the following formula $P_i^{drop} = 20\% \times u_i(b_i^{req}) \times t_i^{avg}$, where b_i^{req} is the requested bandwidth, and t_i^{avg} is the average connection duration (according to Table II). Adaptation time was set to 5 seconds.

As our main performance metric we use the accumulated system utility (u^a) generated by the different connections in the system. The accumulated system utility is independent of the allocation algorithm and is calculated in the same way for all the simulated schemes and according to Section IV.

VII. Evaluation results

Figure 8 presents the accumulated utility generated by 5 allocation schemes (described shortly) during one simulated hour. On the x-axis we have the arrival rate (number of new connections per second). The values in parenthesis represent the corresponding offered load as compared to the capacity of the cell. Thus $0.2(2.56)$ means that the offered load with an arrival rate of 0.2 was 2.56 times the maximum capacity of the cell. The offered load is calculated using the bandwidth requests of the connections.

For each of the arrival rates and for each bandwidth allocation scheme we conducted five different experiments (by changing the seed of the various distributions) and plotted the average value. The coefficient of variance (CV) was less than 0.06 in almost all of the cases ($CV = \sigma/\mu$, that is the standard deviation divided by the average). A similar statistical confidence applies also to the results presented in the forthcoming figures.

A. Comparison to basic maximisation algorithm

To see the impact of our class and age aware modifications, we have compared three flavours of TARA. TARA-normal and TARA-perf-est both use modified R-U functions as presented in Section V. The difference is that for TARA-normal we have used the average connection duration (see Table II) to estimate the duration of each connection when calculating the modifications (see Section V), while for TARA-perf-est we used the real duration from the traffic generator. Thus, the latter provides the best possible case to hope for. Although

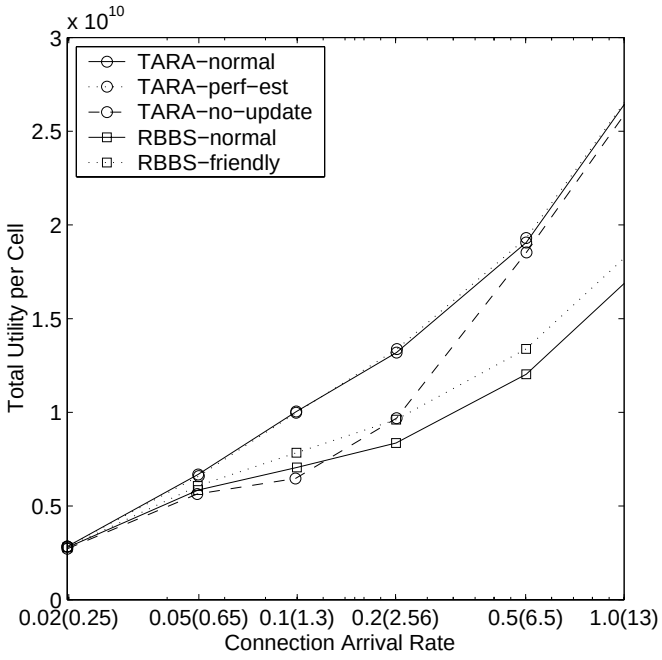

Fig. 8. Accumulated utility

the age-dependent modifications play an important role in our scheme, the difference between TARA-normal and TARA-perf-est in Figure 8 is marginal. It seems that in most of the cases, the difference between the real duration and the average value, is too small to result in the wrong decision (to decisively change the slopes of the modified R-U functions).

We have also simulated a version of TARA where the modifications of the original R-U functions are not performed, denoted as TARA-no-update. Basically, TARA-no-update is the convex_hull_opt allocation algorithm (see Section III-B) invoked periodically. By not taking into consideration the connection classes, the dropping penalty and the adaptation time, TARA-no-update exhibits a 35% decreased system utility when working in areas where the offered load is between 1.3 and 2.6.

At high overloads (corresponding to 0.5 and 1 arrival rate) the applications with the lowest utility per bit, which belong to application group 3, class II, are all rejected at the beginning, and since the lowest utility per bit connections still accepted are now applications in group 6 class III, which can be put indefinitely on hold, TARA-no-update comes closer to the other two. This is an expected behaviour with a traffic in which the allocation borderline (the last bandwidth allocated) lies firmly within connection class III.

B. Comparison with RBBS

The results for RBBS have been plotted as RBBS-normal. There is a large difference between TARA and RBBS which amounts to 45% when the system gets overloaded with traffic. The main factor that contributes to this result is the absence of utility consideration by RBBS. While TARA is rejecting only low utility per bit connections, RBBS is rejecting a comparable amount from all application groups.

Besides the original RBBS we also used a slightly modified version of RBBS to make the comparison more favourable towards that scheme (shown as RBBS-friendly). The original RBBS may both lower and raise bandwidth for all connections. Hence, we modified RBBS not to replenish connections of TARA class II (because no utility is gained), and set the borrowable part of TARA class I connections to zero. For both RBBS schemes, reserved bandwidth was $r = 5\%$, number of levels $\lambda = 10$, and borrowing factor $f = 0.5$ [7].

C. QoS per application group

So far we have presented the results only from the perspective of the total system utility. A more specialised view is presented in Table III, the application groups on the x-axis refer to those in Table II. We can observe that only connections that have the lowest utility efficiency are blocked (new connections) or dropped (ongoing connections). Since application group 6 is a class III connection, it can accept zero allocation situations, so there are no ongoing connections dropped in that case. Also, even at 13 times overload, most of the small, important application groups remain unscathed. Nevertheless it is important to note that the main goal of the system is to generate the highest utility and not to minimise the number of rejected/dropped connections.

TABLE III

STATISTICS PER APPLICATION GROUP AT LOAD 2.42 AND 13

	load = 2.56						load = 13					
application groups	**1**	**2**	**3**	**4**	**5**	**6**	**1**	**2**	**3**	**4**	**5**	**6**
accepted new	464	461	234	445	466	394	2417	2400	66	2346	2355	610
rejected new	0	0	216	0	0	115	0	0	2283	0	0	1917
rejected ongoing	0	0	74	0	0	0	0	0	49	0	0	0
allocation level (%)	100	100	66	100	96	85	100	100	24	100	81	50

D. Choice of performance metric

As the main performance metric, we use the accumulated system utility. Hence, we depart from the traditional call blocking probability (CBP) and call dropping probability (CDP) as performance metrics. We argue that they are obsolete in a system where the requested bandwidth of one connection might be only a small fraction of another connection's demands, but both contribute equally in calculating CBP or CDP. The argument is confirmed by Figure 9, which shows the CBP of the simulations. The application group most blocked by TARA has a big bandwidth demand, and by blocking few of them a lot of bandwidth is saved for other connections. Since RBBS treats all connections equally it has to reject much more connections to equal the number of bits.

Although the aim of our algorithm is to maximise the utility and not to ensure a low dropping (or blocking) probability, dropping an accepted connection reveals a certain degree of miscalculation. Thus we present the CDP in Figure 10. Since TARA can also drop ongoing connections which are not handed over, we use a different formula for CDP.

$$CDP = \frac{rejectedOngoing + rejectedHandovers}{acceptedNew + acceptedHandovers}$$

Even without reserving a certain amount of bandwidth to be used exclusively for handovers (RBBS reserves 5% for this purpose), TARA-normal and TARA-perf-est are able to keep the number of droppings quite low. Handovers are not regarded as new connections in the cell where they are handed over. We want to emphasise here that the increased importance due to the aging mechanism provides a "natural" differentiation between handovers and new connections, and the algorithm does not have to use some kind of forced differentiation mechanism to differentiate between them.

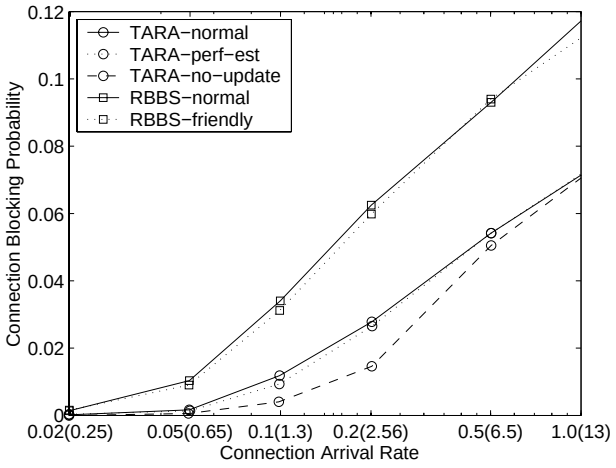

Fig. 9. Connection blocking probability

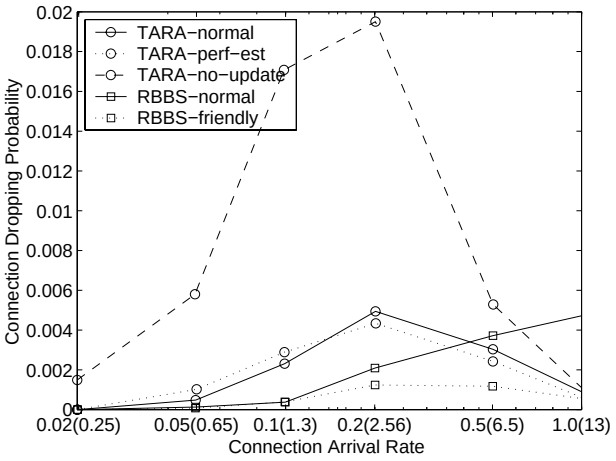

Fig. 10. Connection dropping probability

The experiments show that the aging mechanism, the dropping penalty, and the flexibility of class III connections are able to protect handovers as well as other ongoing connections from being dropped. The consequence of not taking in to consideration these factors is shown in the plot of TARA-no-update. While blocking less connections, it is dropping more than TARA-normal. The effects on the accumulated utility were already presented in Figure 8.

E. Complexity considerations

From a computational complexity point of view, the convex hull maximisation algorithm that we use, has a complexity of $O(nL \log n)$, where n is the number of ongoing and new connections, and L is the maximum number of utility levels of an R-U function. The utility function modifications that we introduce have the complexity of at most $O(nL)$, since they have to manipulate each level in the R-U function. The RBBS

algorithm has a worst case complexity of $O(n)$, since it has to access each connection when borrowing bandwidth. When borrowing does not occur, that is until free bandwidth is depleted, the algorithm just serves the new incoming connections ($O(1)$). The above analysis shows that considering only algorithmic computations, the RBBS has a favourable complexity compared to ours. However, the bandwidth reallocations might impose a heavy burden on the system due to executions of control functions and the associated signalling. Since we expect that the reallocation overhead is more important than the computational complexity, we specifically study the trade-off between utility optimisation and reallocation overhead.

VIII. Bandwidth reallocations and infrastructure overhead

By analysing the applications' requirements and modifying the R-U functions the allocation algorithm takes account of the effect of reallocations from an application point of view. However, the scheme ignores the overheads created by reallocations on the system infrastructure. First, there is the increase in signalling. Every time the allocated bandwidth changes, the radio network controller (RNC), where the radio resource management decisions are taken, has to send the bandwidth reallocation decision to both source and destination of the respective connection. Second, processing capacity is demanded by control functions used in conjunction with a bandwidth change (e.g. channel switching: both type and rate, setup/release connection, power control etc.). A more detailed relationship between control functions and CPU load in a RNC, together with a lower-level CPU overload protection mechanism were presented in an earlier work [14].

To measure the infrastructure overhead we count the number of changes occurring in the respective cell at each reallocation point. Note that the overhead is proportional to the *number* of bandwidth changes and does not depend on the *amount* of bandwidth change. Figure 11 compares the number of bandwidth reallocations for TARA and RBBS for different connection arrival rates (i.e. offered load).

As in the previous graphs, the x-axis depicts the connection arrival rate (CAR) and the values in parenthesis represent the offered load compared to the capacity of the cell. While the system is not overloaded (before 0.1 on the x-axis) the number of changes is low. It begins to grow (approximatively linear) when the system is overloaded. Under initial overload (2.56 cell capacity) the difference between TARA and RBBS is around 50%. The difference slowly decreases as the system gets heavily overloaded with traffic. It is explained by the fact that, RBBS is borrowing bandwidth from all connections in the system (and thus reallocating for all), while TARA is degrading only the lowest performers.

Our next concern is the following: how to modify the allocation maximisation algorithm so that we decrease the number of bandwidth changes while still keeping a high total system-wide utility? Or conversely, what is the relationship between the number of reallocations and the performance (generated utility) of the system?

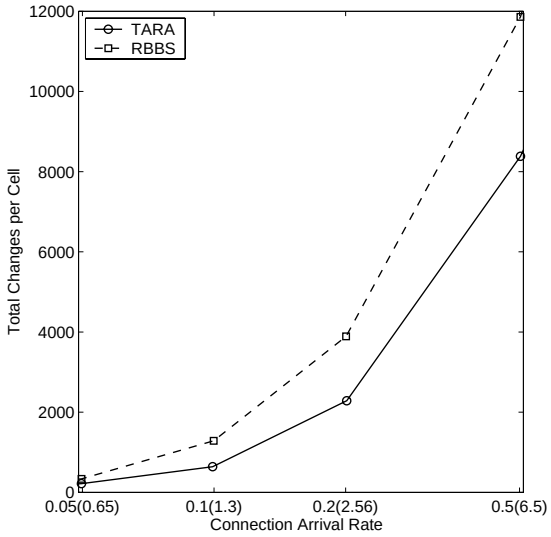

Fig. 11. Number of reallocations comparison

A. Reallocation control

The magnitude of overhead generated by bandwidth reallocations and their effects on the system-wide utility greatly depends on the system implementation and also on the runtime environment. Thus, it is difficult to analytically associate a penalty tag to a reallocation to use it directly in the utility maximisation algorithm. Nevertheless, it is clear that by reducing the number of reallocations the overhead is proportionally reduced. To this end, we propose and evaluate an enhanced bandwidth allocation algorithm that minimises the number of bandwidth changes while keeping the utility at a predetermined level. The algorithm, presented in Figure 12, will replace the original $convex\_hull\_opt$ algorithm in TARA (see the (re)allocation algorithm in Section V-D).

The core idea is as follows: each period we first compute a bandwidth allocation that maximises system utility regardless of the number of generated bandwidth changes (using $convex\_hull\_opt$). This will generate a list of changes to be performed. Then we discard changes from the list as long as the utility remains above a minimal acceptable value. The minimal acceptable utility is proportional to the maximal utility allocation and is calculated using a control threshold parameter ($thresh$). That is, if $thresh = 95\%$, the utility generated in each period is never lower than 95% of the maximum attainable. The pseudocode below describes the algorithm.

Figures 13 and 14 show the performance of the algorithm (measured in the number of reallocations), when used with different threshold settings. All the other parameters used in the experiments are identical to the ones presented in Section VI. Figure 13 presents the dependency between reallocation count and offered traffic load. As the traffic load increases the difference between the allocation algorithms increases too. For instance, when CAR = 0.1 (the offered traffic load is 130%), the number of reallocations with $thresh = 80\%$ is 25% lower than for TARA-normal. With the new algorithm when CAR = 0.5 (the offered load is 650%) the number of reallocations drops with 68%. Note that for a given threshold

MinChangeAlloc Algorithm
input: $\{c_1, c_2, ..., c_n\}$ //set of connections
$thresh$ //system utility threshold
output: $\{b'_1, b'_2, ..., b'_n\}$ //newly allocated bandwidth
begin
initialise: $u_i :=$ R-Ufunction(c_i) //standard R-U function
$b_i := \text{bandwidth}(c_i)$ //current bandwidth
$\{b'_1, b'_2, ..., b'_n\} := \text{convex\_hull\_opt}(u_1, u_2, ..., u_n)$
$u' := u^{max} := \sum_{i=1}^{n} u_i(b'_i)$ //maximum utility
$G := \{\ c_i\ |b'_i > b_i\}$ //connections gaining bw.
$L := \{\ c_i\ |b'_i < b_i\}$ //connections losin bw.
$eff_i := |\frac{u_i(b'_i) - u_i(b_i)}{b'_i - b_i}|$ //the efficiency criterion
$G_a := \text{sort}_a(G)$ //sort efficiency-ascending
$L_d := \text{sort}_d(L)$ //sort efficiency-descending
$fbw := \text{B\_}max - \sum_{i=1}^{n} b'_i$ //calculate initial free bw.
while $G_a \neq \emptyset \wedge L_d \neq \emptyset$ **loop**
//identify a bw. decrease and a set of covering increases:
identify $c_l = \text{head}(L_d)$
if $\exists$ a minimum prefix PG_a of G_a such that
$b_l - b'_l \leq \sum_{c_j \in PG_a} (b'_j - b_j) + fbw$ **then**
//resulting utility if we give up these reallocations:
$u' := u' - \sum_{c_j \in PG_a} (u_j(b'_j) - u_j(b_j)) - u_l(b'_l) + u_l(b_l)$
if $u' < thresh \times u^{max}$ **then exit loop**
//else really give these reallocations up:
$fbw := fbw - \sum_{c_j \in PG_a} (b'_j - b_j) - b'_l + b_l$
$b'_l := b_l;\ \forall c_j \in PG_a\ b'_j := b_j$
$G_a = G_a - PG_a;\ L_d = L_d - \{c_l\}$
else exit loop
end loop
return $\{b'_1, b'_2, ..., b'_n\}$
end

Fig. 12. Reallocation control algorithm

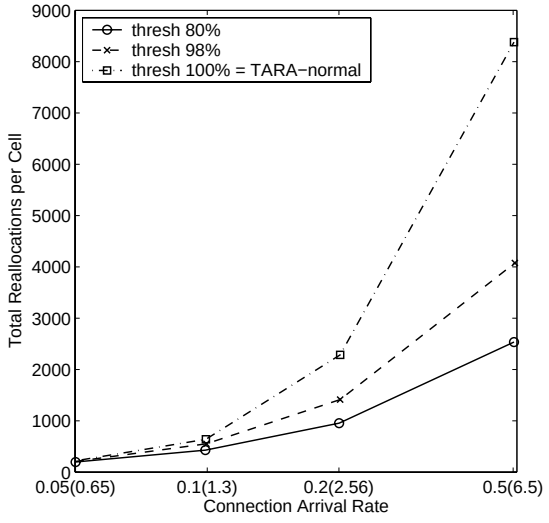

Fig. 13. Performance of reallocation minimisation algorithm. View (a)

value (representing the minimal acceptable utility) the actual utility gained by the system can be much higher. A better perspective on the dependency between reallocations and the generated utility is presented in Figure 14. While for lower loads, the increase in reallocations is not so big, for higher traffic loads (higher connection arrival rates), the number of reallocations strongly increases as we get closer to maximum utility. Thus we can greatly diminish the number of bandwidth

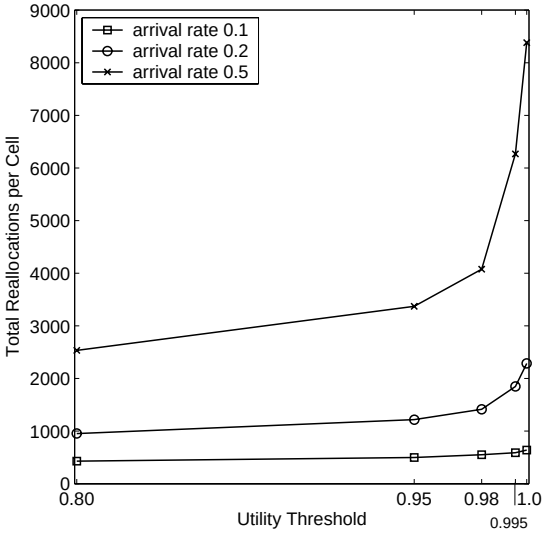

Fig. 14. Performance of reallocation minimisation algorithm. View (b)

reallocations by slightly lowering the utility expectations.

IX. Conclusions

In an open, dynamic system there is a trade-off between optimisation and provisioning. A resource allocation decision might promise maximal utility at a certain time point, but as new requests arrive and old requests depart, a reallocation could improve the utility of the system. The question arises: should resources be reallocated or not? A reallocation might accept connections with higher utility, but might also break ongoing QoS contracts. The novelty of our approach is that we combine these choices in a consistent manner. We synthesise the consequences of potential reallocations for different classes of applications, and use this information in our periodic allocation/reallocation strategy.

We have presented an admission control and resource allocation scheme to be used in a future generation mobile network. The scheme is based on an allocation algorithm that aims to maximise system-wide utility, having the utility of each connection specified by a bandwidth dependent utility function. To suit the dynamic nature of the environment, where constant reallocations are required, we identified the effects of reallocations on different connections. Based on their sensitivity to reallocations, connections have been divided into three classes: non-flexible, semi-flexible, and fully flexible. Connections are also differently affected by disconnections and have different sensitivity to allocation fluctuations. An implicit but very important factor is the age of the connection, since it represents the time during which resources have already been invested.

While the application here might seem too specific, we believe that a similar approach can be adapted for other open, dynamic environments (e.g. the link capacity of an Internet provider) or other resource types (e.g. power-aware computing, or bottlenecks in adhoc networks).

To validate our approach, the algorithm has been tested against a baseline that does not take account of the above factors. We have also compared it with a recent adaptive

allocation scheme (RBBS), that does not use a value-based approach. Our approach shows significantly increased performance as expressed by the system-wide accrued utility. Another advantage is that the treatment of handovers is consistent with that of other ongoing connections, by taking into account their age-related increased importance when allocating bandwidth in the new cell.

After identifying a (re)allocation scheme based solely on application preferences, we considered the overhead that the scheme generates in terms of demands on other system-wide resources. For example CPU time utilisation and signalling traffic will increase when executing the reallocation decisions. We have chosen the number of bandwidth reallocation as a metric for characterising such demands and shown that we can greatly reduce the strain on the system with only a small decrease in the generated utility. Consequently, this new algorithm could be employed to avoid overloading the infrastructure.

An immediate step as a future work is by extending the setting to a multi-link resource allocation. Hybrid access networks are very interesting for they combine the cellular hierarchy with ad-hoc flexibility. Besides the topological challenge there is also the resource difference. Even the channel conditions are usually different, for instance ad-hoc channels are more unreliable, and this will affect their QoS. A consistent, multi-resource allocation optimisation scheme is a longer-term goal.

We conclude by making the following remark. In a future generation mobile network, the resources (bandwidth) required by different applications and services will be highly varied. Blocking or dropping connections will be too coarse-grained to be suitable for such a situation, The concern ought to shift on how often and by how much resources ought to be (re)allocated. Thus, using a performance metric such as the accumulated system utility will enable a better control over system behaviour.

Acknowledgements

This work has been supported by the Swedish National Graduate School in Computer Science (CUGS) and Center for Industrial Information Technology (CENIIT) at Linköping University. The authors also wish to thank Robert Jonasson who implemented the simulator and Mona El-Kadi for providing valuable information about the RBBS scheme. The comments by anonymous reviewers were helpful in improving the presentation of the paper.

References

[1] T. F. Abdelzaher, K. G. Shin, and N. Bhatti. Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Trans. Parallel Distrib. Syst.*, 13(1):80–96, 2002.

[2] C. Aurrecoechea, A. T. Campbell, and L. Hauw. A survey of qos architectures. *Multimedia Systems Journal, Special Issue on QoS Architecture*, 6(3):138–151, May 1998.

[3] V. Bharghavan, K.-W. Lee, S. Lu, S. Ha, J.-R. Li, and D. Dwyer. The timely adaptive resource management architecture. *Personal Communications, IEEE*, 5(4):20–31, aug 1998.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services, rfc 2475. RFC 2475, Dec. 1998.

[5] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview, rfc 1633. RFC 1633, June 1994.

[6] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (rsvp), rfc 2205. RFC 2205, Sept. 1997.

[7] M. El-Kadi, S. Olariu, and H. Abdel-Wahab. A rate-based borrowing scheme for qos provisioning in multimedia wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(2):156–167, Feb. 2002.

[8] http://javasim.cs.uiuc.edu. Javasim home page.

[9] R. Jonasson. Simulator for resource allocation in future mobile networks. Master's thesis, Linköping University, Oct. 2002.

[10] C. Lee. *On Quality of Service Management*. PhD thesis, Carnegie Mellon University, Aug. 1999. Technical Report CMU-CS-99-165.

[11] C. Lee, J. Lehoczky, R. Rajkumar, and D. Siewiorek. On quality of service optimization with discrete qos options. In *Proceedings of the IEEE Real-time Technology and Applications Symposium*, June 1999.

[12] R. R.-F. Liao and A. T. Campbell. A utility-based approach for quantitative adaptation in wireless packet networks. *Wireless Networks*, 7:541–557, Sept. 2001.

[13] X. Liu, Q. Wang, and L. S. W. He. Optimal qos sampling frequency assignment for real-time wireless sensor neworks. In *Proceedings of the 24th Real-Time Systems Symposium*, pages 308–319. IEEE Computer Society, dec 2003.

[14] S. Nadjm-Tehrani, K. Najarian, C. Curescu, T. Lingvall, and T. A. Dahlberg. Adaptive load control algorithms for 3rd generation mobile networks. In *Proceedings of the 5th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 104–111. ACM CS-press, Sept. 2002.

[15] C. Oliveira, J. B. Kim, and T. Suda. An adaptive bandwidth reservation scheme for high-speed multimedia wireless networks. *IEEE Journal on Selected Areas in Communications*, 16:858–878, Aug. 1998.

[16] P. Richardson, L. Sieh, and A. Ganz. Quality of service support for multimedia applications in third generation mobile networks using adaptive scheduling. *Real-Time Systems*, 21(3):269–284, Nov. 2001.

[17] F. Ruben and F. Edlund. Design of a mobile payment intermediary. Master's thesis, Linköping University, LiTH-IDA-Ex-02/X, Dec. 2002.

[18] H.-Y. Tyan and C.-J. Hou. Javasim : A component-based compositional network simulation environment. In *Western Simulation Multiconference - Communication Networks and Distributed System Modeling and Simulation*, June 2001.

Calin Curescu received his BSc degree from "Politehnica" University of Timisoara, Romania in 1999 and Licentiate degree from Linköping University, Sweden in 2003. He is currently a PhD candidate in the Department of Computer and Information Science at Linköping University. His research interests include quality of service in dynamic distributed systems such as wireless networks, real-time systems, QoS based routing, and utility based performance metrics.

Simin Nadjm-Tehrani is associate professor and director of Real-time Systems Laboratory (RTSLAB) at Department of Computer and Information Systems, Linköping University, Sweden. She obtained her BSc from Manchester University, England, and her PhD degree from Linköping University. Her recent research interest is in the area of dependability for distributed real-time systems, including analysis of safety and fault tolerance, and quantified system availability in presence of failures, attacks and overloads