Flow-level modeling and analysis of dynamic TDD in LTE

Pasi Lassila, Aleksi Penttinen, Samuli Aalto
Aalto University School of Electrical Engineering, Otakaari 5, FI-02150, Finland
Email: firstname.lastname@tkk.fi

***Abstract*—We present a queueing analysis of elastic traffic performance in LTE systems using the dynamic TDD scheme. Both fair resource sharing and performance optimization using different approaches are considered. We first analyze the system without any restrictions in the resource allocation between the uplink and downlink and demonstrate that a simple dynamic scheduling scheme (called here dynamic-PS), where the allocation for a single flow is always inversely proportional to the total number active flows, shows good performance and fairness properties compared with any optimized static allocation scheme. We also consider the achievable gains with more detailed traffic statistics, including the application of the Gittins index policy and SRPT. However, the actual LTE TDD system only supports a discrete set of possible allocations in the capacity region. We then investigate how these allocation constraints impact the performance of the discretized variant of the dynamic-PS policy by using different approaches. To optimize the performance we apply MDP for exponential service times and, for example, derive a structural result that the optimal policy always selects among two corner points of the capacity region. Also, an SRPT-like heuristic scheduling algorithm is given. The analytical and simulated results suggest that the discrete dynamic-PS policy is robust against impact of different service time distributions, fair and performs reasonably well.**

I. Introduction

Modern cellular systems (LTE) allow the radio resources to be scheduled very flexibly. The scheduling interval is also very short (in the order of milliseconds) and the scheduler has state information allowing the selection of the coding and modulation to match the state of the users' channels. Also, the 3GPP standards for LTE specify the possibility of using the same frequency band for transmitting both the uplink (UL) and downlink (DL) traffic, i.e., the so-called TDD (time division duplexing) mode. The dynamic TDD scheme refers to the possibility of making the adaptation dynamically based on traffic conditions, which can, e.g., lead to improved downloading performance for elastic TCP file transfers [1].

Performance of elastic traffic in a given system manifests itself at the so-called *flow level* (with a much longer time scale compared to the scheduling interval), where the system must be considered in a dynamic setting with random-sized flows arriving stochastically. Flow-level models for cellular systems typically focus on only downlink traffic or uplink traffic, see, e.g., [2]–[5]. The performance of time-slot level resource allocation in dynamic TDD has been considered recently, e.g., in [1], [6], [7]. However, these studies fail to account for the inherently random nature of the active user population.

In this paper we model and analyze the flow-level performance of elastic data traffic in a cellular system operating under dynamic TDD. To gain preliminary insights to the problem, we first consider the *simplified case* where the allocation of time between the two classes (UL and DL) can be done arbitrarily, which leads to standard (multi-class) M/G/1 models. We consider two fair policies which correspond to PS-models at the flow level. In the static-PS policy, the time allocation is fixed between the classes but can be optimized. We note that this optimization has been recently considered in [8] for a system with both streaming and elastic flows. The static-PS policy also represents the currently considered approaches in 3GPP standardization [1]. However, the optimal static policy can be improved by making the division of time more dynamic so that the allocation is always proportional to the number of active flows in the classes. This constitutes the dynamic-PS policy and it can be shown to have always a better performance than the static-PS policy. To further optimize the mean delay performance, we introduce the optimal distance-aware non-anticipating policy, based on the Gittins index policy [9]–[11], as well as the globally optimal SRPT policy [12]. For the special case with exponential file sizes, we show that the Gittins index policy corresponds to a specific priority policy for which the mean delay analysis can be done analytically. The Gittins index policy provides one point of comparison for what can be achieved with more detailed information about the flows and the optimal SRPT policy provides the absolute lower bound. Our main finding here is that the dynamic-PS policy represents a practical, simple and robust approach showing good performance and fairness properties compared with the static-PS policy. Also, the results demonstrate that the benefits of having SRPT-like knowledge of remaining service times are considerable, although at the expense of increased unfairness between UL/DL performance.

In practice the allocation of time between the two classes cannot be done in an arbitrary manner but the system is constrained to a small set of possible allocations, see [1]. In this *more realistic case* with the allocation constraints, the problem cannot any longer be mapped to a standard M/G/1 queue and no theoretical results are available for determining the optimal performance. The region of feasible time allocations can be interpreted as the *capacity region*, and our problem is similar to issues already analyzed for bandwidth sharing in wired and wireless networks with different capacity regions [13], [14]. Motivated by the robust properties of dynamic-

PS in the unconstrained case, we first present the discrete variant of the dynamic-PS policy. In addition, we apply the MDP (Markov Decision Processes) theory [15] to optimize the scheduling by policy iteration under exponential service times and knowledge of only the number of flows in the classes. We obtain an interesting structural result that the optimal policy always uses either of the extreme points (corner points) in the allocation region. Third, we apply results for Balanced Fairness in order to obtain an insensitive bound for the performance that approximates the discrete dynamic PS-policy [13]. Here we utilize the fact that the capacity region of our system can be interpreted as the same as for a tree network with two access links and one common link having fixed capacities. Finally, we develop an SRPT-like heuristic that is based on the obtained insights. Our results indicate that the performance of the discrete dynamic-PS policy is close to that of the unconstrained dynamic-PS and it cannot be much improved by purely knowing the number of flows. Only after having detailed SRPT-like information, the performance can still be significantly improved. Thus, the discrete dynamic-PS policy is indeed a practical approach for realizing flow-level scheduling in a dynamic TDD system.

Note that we are in this paper focusing on the case, where the scheduler does not utilize instantaneous channel state information. Flow-level analysis of channel aware schedulers is difficult as the capacity region becomes state dependent. Stability of such schedulers has been considered in [16] and optimization jointly with size-based information poses a challenging trade-off, see [17], [18].

The paper is organized as follows. The model is introduced in Section II. In Section III we consider the system without the allocation constraints and obtain preliminary insights to the problem. Section IV contains the analysis of the constrained system. Conclusions are given in Section V.

II. The system model

Consider a single base station that is serving the users within its coverage area. In the model, we will jointly consider both uplink and downlink transmissions from these users. The traffic consists of elastic flows representing, e.g., file transfers. Each flow is associated with a particular user, and thus subsequently we may use the words user/flow interchangeably.

The uplink and downlink flows are represented as classes, and in each class, flows arrive according to a Poisson process with rate λ_u for the uplink class and with rate λ_d for the downlink class, respectively. To characterize the random variables for service times of the classes, denoted by S_u (uplink) and S_d (downlink), we use a similar spatial model as in [2], [19]. Users are distributed inside the cell according to a spatial Poisson process. Consider a user, say from the uplink class, with a random size X_u and at a random distance R (which is independent of X_u). The service time S_u is defined as $X_u/c_u(R)$, where the function $c_u(r)$ represents the mean achievable rate of a user at distance r (when scheduled). The service time for the downlink class, S_d, is defined in the same way. The generic capacity function $c(r)$ for the rate at distance r can be, for instance, defined as

$$c(r) = \min\{C_0, (r_0/r)^\alpha\}, \tag{1}$$

where C_0 is the maximum achievable rate within a disk of radius r_0 around the base station, and outside of this, the rate decays according to the path loss exponent α outside. The parameters C_0, r_0 and α can be different for the uplink and downlink classes, resulting in different capacity functions for the uplink and the downlink. Thus, we assume a time scale separation between the time slot level and the flow level, as in [2]–[5], [19]. Typically, the service times S_u and S_d do not have identical distributions, i.e., they are asymmetric.

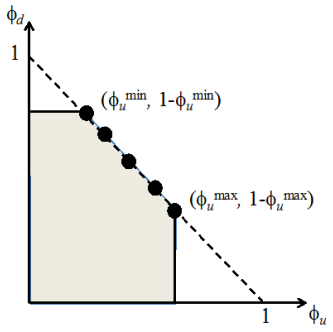

Fig. 1. The region of feasible allocations in the dynamic TDD system.

Let (ϕ_u, ϕ_d) denote the fraction of time allocated to the uplink and the downlink flows, respectively. The discrete set of possible (time slot level) allocations of dynamic TDD in LTE are indicated by black circles in Figure 1. Let us denote this set of possible allocations for the uplink class by $\Phi_u = \{\phi_{u,1}, \ldots, \phi_{u,K}\}$, where it is assumed that $\phi_{u,1} < \ldots < \phi_{u,K}$. Furthermore, we denote by $\phi_u^{\min} = \phi_{u,1}$ and $\phi_u^{\max} = \phi_{u,K}$. Of special importance are the corner points, denoted by $(\phi_u^{\min}, 1-\phi_u^{\min})$ and $(\phi_u^{\max}, 1-\phi_u^{\max})$. For the dynamic TDD in LTE, $K = 5$, $\phi_u^{\min} = 1/10$, and $\phi_u^{\max} = 3/5$ [1]. The colored area in Figure 1 (including all the points along the line $\phi_u + \phi_d = 1$ with $\phi_u \in [\phi_u^{\min}, \phi_u^{\max}]$) gives the capacity region at the flow level, which can be realized by time sharing at a faster time scale. The entire area under the dashed line, representing $\phi_u + \phi_d = 1$, corresponds to the set of feasible allocations without any constraints.

The load of uplink flows, ρ_u, is given by $\rho_u = \lambda_u \mathrm{E}[S_u]$ and similarly the load of downlink flows equals $\rho_d = \lambda_d \mathrm{E}[S_d]$. A necessary condition for the system to be stable is

$$\rho = \rho_u + \rho_d < 1, \quad \rho_u < \phi_u^{\max}, \quad \text{and} \quad \rho_d < 1 - \phi_u^{\min}.$$

However, depending on the policy and the distribution of the service time requirements this may not be sufficient. Furthermore, the performance can be optimized with an appropriate scheduling policy.

III. Analysis of unconstrained dynamic TDD

Here we discuss the simplified case where we assume that the time allocation (ϕ_u, ϕ_d) can be done in an unconstrained manner, i.e., it can be selected freely along the line $\phi_u + \phi_d = 1$

in Figure 1. Under this assumption, the system corresponds to a standard M/G/1 queue with two traffic classes and existing results can be applied to analyze the system. To study the performance of the system, our main focus is on minimizing the mean flow delay by scheduling.

A. Scheduling under fair sharing

Here we consider simple fair policies, which offer a reasonable practical approach for realizing the scheduling in the system.

Optimal static-PS: In the static-PS policy, the time allocation between the classes is defined in a static manner, and the base station is assumed to know the loads of the classes ρ_u and ρ_d. Additionally, it is assumed that within each class the scheduler at the time-slot level shares the resources in a round-robin manner, which at the flow level can be modeled as a *processor-sharing* (PS) queue [20]. Since the time allocation between the classes is static, the system reduces to two independent M/G/1 PS queues. It is straightforward to minimize the mean number of flows $\mathrm{E}[N] = \rho_u/(\phi_u - \rho_u) + \rho_d/(1 - \phi_u - \rho_d)$ with respect to ϕ_u, which yields as the optimal resource sharing $\phi_u = (\sqrt{\rho_u} + \rho_u\sqrt{\rho_d} - \sqrt{\rho_u}\rho_d)/(\sqrt{\rho_d} + \sqrt{\rho_u})$ and the performance of the optimal static-PS policy

$$\mathrm{E}[N^{\mathrm{S-PS}}] = \frac{\left(\sqrt{\rho_d} + \sqrt{\rho_u}\right)^2}{1-\rho}.$$

The performance is also insensitive to the service time distributions. This also represents a practical comparison policy for our study, as it represents the currently considered approaches in 3GPP standardization, see [1].

Dynamic-PS: To improve the static policy, in the dynamic-PS policy the base station allocates dynamically the fraction of time between each class to be proportional to the number of flows in the class, i.e., $\phi_u = N_u/(N_u + N_d)$ with N_u and N_d denoting the number of uplink and dowlink flows. Within each class users are served according to PS. In this case, the total number of customers evolves as in a single class M/G/1 PS queue with joint arrival rate $\lambda_u + \lambda_d$ and the service time is distributed according to S_u or S_d with probabilities $\lambda_u/(\lambda_u + \lambda_d)$ and $\lambda_d/(\lambda_u + \lambda_d)$, respectively. Thus, the mean number of flows in the system under the dynamic-PS policy, $\mathrm{E}[N^{\mathrm{D-PS}}]$, is given by

$$\mathrm{E}[N^{\mathrm{D-PS}}] = \frac{\rho}{1-\rho}.$$

Again, the performance is insensitive to the service time distribution. In addition, it is easy to see that $\mathrm{E}[N^{\mathrm{D-PS}}] < \mathrm{E}[N^{\mathrm{S-PS}}]$ for any stable load.

Note that the dynamic-PS policy only requires knowledge about the number of active flows, and based on that, dynamically adapts the service. Also, this information may actually be easier to estimate than the loads of the classes, as is required in the static-PS policy. However, the drawback is that the allocation changes as the number of active flows varies randomly, i.e., the time scale of scheduling decisions is faster than in the static-PS policy.

B. Optimization of the mean delay

The above PS policies are simple and assume relatively little information about the traffic. To optimize the performance it is possible to apply size-based scheduling where the underlying idea is to favor smaller flows in order to efficiently minimize the number of flows in the system, which also minimizes the mean flow delay. Next we consider two such approaches.

Gittins index: First we consider the case, where the base station only has information on how the service of the flows has progressed up to time t, i.e., information on how much is remaining is not available. Hence, we speak about so-called non-anticipating policies. Within this family of policies, the *Gittins index* approach yields the optimal policy for minimizing the mean flow-delay in the M/G/1 queue with multiple classes [9]–[11].

The Gittins index (GI) policy is a preemptive policy where the scheduler always serves the flow with the highest Gittins index. The general form of the policy is rather implicit, see, e.g. [19] for its definition. It requires knowledge of the attained service from each flow and it depends on the exact distribution of the service times S_u and S_d. By definition the service time for uplink flows at distance R is $S_u = X_u/c_u(R)$, and correspondingly for the downlink flows, $S_d = X_d/c_d(R)$. In addition to the notion of uplink/dowlink flows representing classes, the random distance R of a user forms a continuous valued class index for the flows, i.e., we can consider an M/G/1 queue where the class is defined by information about whether the flow is uplink/downlink and its distance. The Gittins index then allows us to obtain the optimal distance-aware non-anticipating policy.

Unfortunately, for general flow size distributions of X_u and X_d there are no analytical results available for the performance. However, if we assume that X_u and X_d obey *exponential distributions* with mean values $\mathrm{E}[X_u]$ and $\mathrm{E}[X_d]$, respectively, the Gittins index is simply the inverse of the mean service time, i.e., $1/\mathrm{E}[S_{u,r}] = c_u(r)/\mathrm{E}[X_u]$ for an uplink flow at distance r, and correspondingly $1/\mathrm{E}[S_{d,r}] = c_d(r)/\mathrm{E}[X_d]$ for a downlink flow at distance r. Thus, Gittins index policy becomes a greedy priority policy with respect to the service rate of the flow, which is also referred to as the *$c\mu$-rule*, cf. [21]. By applying the well known results for priority queues [20], we conclude that the mean delay for an uplink flow with the receiving terminal located at distance r from the base station is as follows:

$$\begin{aligned}\mathrm{E}[T^{\mathrm{GI}}_{u,r}] &= \frac{\mathrm{E}[S^*_{u,r}]}{(1-\sigma_{u,r})^2} + \frac{\mathrm{E}[S_{u,r}]}{1-\sigma_{u,r}}, \\ &= \frac{\mathrm{E}[S^*_{u,r}]}{(1-\sigma_{u,r})^2} + \frac{\mathrm{E}[X_u]}{c_u(r)(1-\sigma_{u,r})}.\end{aligned}$$

Here, $\sigma_{u,r}$ is the expected load seen by an uplink flow at distance r, and it depends on the uplink flows up to distance r and downlink flows up to distance $s(r)$ defined as

$$s(r) = \sup\{x : \mathrm{E}[S_{d,x}] < \mathrm{E}[S_{u,r}]\},$$

i.e., all these uplink and downlink flows have a higher priority than the considered uplink flow at r. The expression for $\sigma_{u,r}$

reads as

$$\sigma_{u,r} = \lambda_u \int_0^r \mathrm{E}[S_{u,z}] dF_R(z) + \lambda_d \int_0^{s(r)} \mathrm{E}[S_{d,z}] dF_R(z),$$
$$= \lambda_u \mathrm{E}[X_u] \int_0^r \frac{dF_R(z)}{c_u(z)} + \lambda_d \mathrm{E}[X_d] \int_0^{s(r)} \frac{dF_R(z)}{c_d(z)},$$

where $dF_R(z) = 2z\,dz$, and S_r^* refers to the so called remaining service time for terminals up to distance r with mean

$$\mathrm{E}[S_{u,r}^*] = \frac{\lambda_u}{2} \int_0^r \mathrm{E}[S_{u,z}^2] dF_R(z) + \frac{\lambda_d}{2} \int_0^{s(r)} \mathrm{E}[S_{d,z}^2] dF_R(z),$$
$$= \lambda_u \mathrm{E}[X_u]^2 \int_0^r \frac{dF_R(z)}{c_u^2(z)} + \lambda_d \mathrm{E}[X_d]^2 \int_0^{s(r)} \frac{dF_R(z)}{c_d^2(z)}.$$

The mean delay of uplink flows is given by

$$\mathrm{E}[T_u^{\mathrm{GI}}] = \int_0^1 \mathrm{E}[T^{\mathrm{GI}}(u,r)]\, dF_R(r).$$

Correspondingly, we have similar equations for the conditional delay of the downlink flows $\mathrm{E}[T_{d,r}^{\mathrm{GI}}]$. The overall performance expressed as the mean number of flows in the system is simply $\mathrm{E}[N^{\mathrm{GI}}] = \lambda_u \mathrm{E}[T_u^{\mathrm{GI}}] + \lambda_d \mathrm{E}[T_d^{\mathrm{GI}}]$.

While the Gittins index policy requires very detailed knowledge, some of which certainly can not be known with very good accuracy, for example, the distribution of the mean service rates over different distances, it serves as a useful bound against which the more robust PS policies can be compared.

SRPT: Finally, assume that the base station has exact information about the remaining service time of each flow, which implies that the base station knows both the remaining size in bits as well as the service rate (i.e., the location of the user in our model). Then the system is represented by a standard M/G/1 queue (as in the case with dynamic-PS) for which *shortest-remaining-processing-time* (SRPT) is the optimal policy for minimizing the mean flow delay [12].

Similarly as in the case of the dynamic PS policy, the system behaves as an M/G/1 queue with a joint arrival rate $\lambda_u + \lambda_d$ and the density of the service time distribution, $f(t)$, is the weighted combination of the uplink and dowlink service time densities, $f_u(t)$ and $f_d(t)$,

$$f(t) = \frac{\lambda_u}{\lambda_u + \lambda_d} f_u(t) + \frac{\lambda_d}{\lambda_u + \lambda_d} f_d(t).$$

It is easy to see that

$$f_u(t) = \int_0^1 f_{X_u}(c_u(r)t) c_u(r) f_R(r) dr,$$

and correspondingly for $f_d(t)$.

The conditional mean delay formula for SRPT originates from [22]:

$$\mathrm{E}[T^{\mathrm{SRPT}}(t)] = \frac{\lambda \mathrm{E}[(S \wedge t)^2]}{2(1-\rho(t))^2} + \int_0^t \frac{1}{1-\rho(s)}\, ds,$$

where $S \wedge t = \min\{S, t\}$ and $\rho(t)$ refers to $\rho(t) = \lambda \int_0^t s f(s)\, ds$. The mean delay of uplink flows is then given by

$$\mathrm{E}[T_u^{\mathrm{SRPT}}] = \int_0^\infty \mathrm{E}[T^{\mathrm{SRPT}}(t)] f_u(t) dt,$$

and similarly for the mean delay of downlink flows $\mathrm{E}[T_d^{\mathrm{SRPT}}]$. Finally, the total mean number of flows is obtained from $\mathrm{E}[N^{\mathrm{SRPT}}] = \lambda_u \mathrm{E}[T_u^{\mathrm{SRPT}}] + \lambda_d \mathrm{E}[T_d^{\mathrm{SRPT}}]$.

Again, the assumptions of the SRPT policy may be somewhat restrictive in practice (complete knowledge of the remaining sizes in bits as well as the service rate of a given flow), but it serves as the absolute lower bound in terms of the mean delay under any scheduling discipline.

C. Comparison of the policies

Here we illustrate the analytical results and compare the different scheduling policies. The parameter values are chosen to reflect realistic system values. As the capacity function for the flows we use the simple form (1). The cell radius is assumed to be 100 m and the radius inside which the maximum rate is achieved is 10 m, and in our unit circle model this corresponds to having $r_0 = 10/100 = 0.1$. Additionally, path loss is $\alpha = 3$. Both r_0 and α were the same for both uplink and downlink. In LTE systems the uplink and downlink rates are asymmetric and thus we use in the downlink the maximum rate $C_d = 200$ Mbps and in the uplink the maximum rate $C_u = 100$ Mbps. The traffic parameters are also asymmetric with respect to the file sizes so that in the downlink the sizes have mean $\mathrm{E}[X_d] = 200$ KB and in the uplink $\mathrm{E}[X_u] = 20$ KB. This reflects the fact that users are typically downloading larger flows/files than uploading. Correspondingly, this gives for the mean service time requirements $\mathrm{E}[S_u] = 0.66$ s and $\mathrm{E}[S_d] = 3.28$ s, respectively, i.e., the asymmetry in the service time requirement equals 5. For the arrival rates we consider two settings. In the first one, the arrival rate of flows in the uplink and downlink are equal, $\lambda_u = \lambda_d$, reflecting roughly that every TCP file download results actually in a bidirectional flow. To increase the load, the arrival rate is increased uniformly. In the other case, the arrival rates are asymmetric in order to see the impact of this. Specifically, we fix $\rho_u = 0.2$ (giving $\lambda_u = 0.31$) and vary ρ_d.

Of main importance for our insights are how the policies perform compared with each other. Thus, for the overall performance we consider the ratio of the mean number of customers relative to the dynamic PS policy, i.e., the ratios $\mathrm{E}[N^{\mathrm{S-PS}}]/\mathrm{E}[N^{\mathrm{D-PS}}]$, $\mathrm{E}[N^{\mathrm{GI}}]/\mathrm{E}[N^{\mathrm{D-PS}}]$ and $\mathrm{E}[N^{\mathrm{SRPT}}]/\mathrm{E}[N^{\mathrm{D-PS}}]$ (due to Little's result, the ratio of the mean number of flows is equal to the ratio of the mean delays). This is depicted as a function of the load ρ in Figure 2 for the different policies (S-PS = optimal static-PS, D-PS = dynamic-PS, GI = Gittins). In the figure, the solid lines correspond to the symmetric arrival rate scenario, while the dashed lines are the results of the asymmetric arrival rate scenario. From the figure, we observe that the dynamic-PS policy is able to perform remarkably well; it gives a

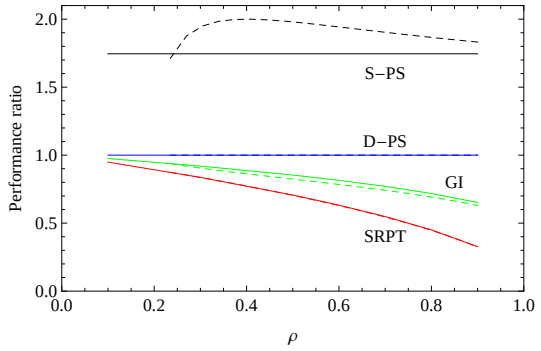

Fig. 2. The relative performance with respect to the dynamic-PS policy for the different policies. The solid and the dashed lines correspond to the results for the symmetric and asymmetric scenarios, respectively.

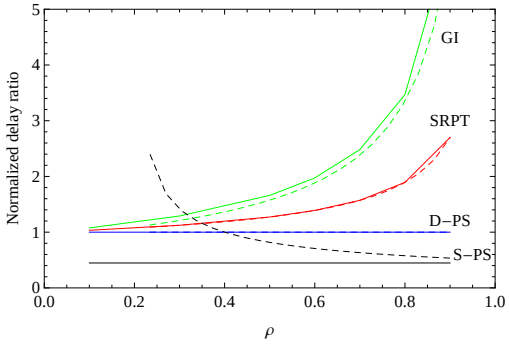

Fig. 3. The ratio of the normalized downlink delay to the normalized uplink delay for different policies as a function of load ρ. The solid and the dashed lines correspond to the results for the symmetric and asymmetric scenarios, respectively.

significant improvement over the static-PS approach, in fact by a factor that equals $(\sqrt{\rho_d}+\sqrt{\rho_u})^2/(\rho_d+\rho_u) > 1$. Note that in the symmetric arrival rate scenario, this becomes a constant, independent of the arrival rate. Also, the dynamic-PS policy with very limited information (number of flows in uplink and downlink) is still able to perform reasonably well against the Gittins index policy that itself depends on quite specific assumptions. Nevertheless, the performance gap to the optimal SRPT policy increases quickly with the load showing the potential usefulness of exploiting size information. These conclusions hold for both the symmetric and the asymmetric arrival rate scenarios; in fact, only the static PS result is clearly affected, while for the Gittins index case it can be barely seen and for SRPT the results are virtually the same.

Next we investigate the fairness properties of the policies. To this end we consider the ratio of the normalized uplink and downlink mean delays $\mathrm{E}[T_d]/\mathrm{E}[T_u] \cdot \mathrm{E}[S_u]/\mathrm{E}[S_d]$, i.e., the uplink mean delay is normalized with respect to the uplink mean service time (and similarly for the downlink case). This is shown for the different policies in Figure 3, where the solid lines and dashed lines correspond to the symmetric and asymmetric arrival rate scenarios, respectively. As can be seen, for both PS policies the ratio remains a constant indicating that the fairness between uplink and downlink is unaffected by load. The D-PS policy is in some sense ideally fair while the static-PS policy favors the downlink class somewhat. The Gittins index policy is highly unfair with respect to the downlink due to its inherent strict priority mechanism (uplink has a smaller service time requirement). SRPT is not quite as bad, but still somewhat penalizes the downlink flows and the unfairness increases with load. Again, only the static PS policy is affected by the symmetric/asymmetric arrival rate setting, while the others (Gittins index and SRPT) are hardly affected at all.

IV. Dynamic TDD with allocation constraints

Here we analyze the impact of having practical constraints in the time allocation between the classes. Under this assumption, the system no longer corresponds to a standard M/G/1 queue and the analysis becomes more challenging. We present the discrete D-PS policy, analyze its performance and consider the performance optimization problem using different techniques.

A. Discrete D-PS policy

We consider the setting where the scheduler can only select the allocation from the discrete set of allocations, see Figure 1. Recall that the set of feasible allocations for the uplink is denoted by Φ_u. Previously, in the unconstrained case, we observed that the D-PS policy offers a good tradeoff between performance and fairness. Additionally, it is simple requiring very little knowledge about the traffic. The discrete D-PS policy simply rounds the ideal allocation to the closest realizable allocation in the set Φ_u. More exactly, let $\phi_u^{\text{D-PS}} = N_u/(N_u+N_d)$, i.e., the ideal allocation according to D-PS policy. The uplink allocation ϕ_u is given by

$$\phi_u = \phi_{u,n*},$$

where

$$n^* = \arg\min\{|\phi_{u,n} - \phi_u^{\text{D-PS}}|,\ n = 1,\dots,K\}.$$

Note that under this policy the system loses its nice insensitivity property. Thus, below we analyze the performance separately with exponential service times and general service times.

Exponential service times: Consider first the case where the service times S_u and S_d obey an exponential distribution. The system under the discrete D-PS policy corresponds to a 2-dimensional Markov process with the transition rates

$$\begin{cases} (N_u,N_d) \to (N_u+1,N_d) : \lambda_u, \\ (N_u,N_d) \to (N_u,N_d+1) : \lambda_d, \\ (N_u,N_d) \to (N_u-1,N_d) : \phi_u/\mathrm{E}[S_u], \\ (N_u,N_d) \to (N_u,N_d-1) : (1-\phi_u)/\mathrm{E}[S_d]. \end{cases}$$

The steady state distribution of the process can be then obtained using standard numerical methods.

General service times: The above analysis specifically relied on Markovian assumptions. Balanced Fairness (BF) resource sharing scheme offers a tractable approach to evaluating the performance of fair sharing under general service

time requirements, see [13]. In our case, we can make the following interpretation of the TDD system. First assume that, in Figure 1, all time allocations along the line $\phi_u+\phi_d = 1$ with $\phi_u \in [\phi_u^{\min}, \phi_u^{\max}]$ can be realized (by using fast scheduling between the discrete points). Then the system under BF can be represented by a tree network with fixed capacities containing two access links with capacities $1-\phi_u^{\min}$ (downlink class) and $\phi_u^{\max}$ (uplink class) and one shared link with unit capacity.

The explicit expression of the normalization constant for the above two-branch tree network originates from [23] and reads

$$G(\rho_u, \rho_d) = \frac{1}{1-\rho_u-\rho_d}\left(\frac{1-\rho_u}{1-\frac{\rho_u}{\phi_u^{\max}}} + \frac{1-\rho_d}{1-\frac{\rho_d}{1-\phi_u^{\min}}} - 1\right).$$

Then the mean number of uplink flows is given by

$$\mathrm{E}[N_u] = \frac{\rho_u}{G(\rho_u,\rho_d)}\frac{\partial G(\rho_u,\rho_d)}{\partial \rho_u},$$

and similarly for the mean number of downlink flows. These results can be used to obtain approximations of the impact of general service time distributions on the fair sharing of the capacity region, such as provided by the discrete D-PS policy in the real TDD system with allocation constraints.

B. Optimizing the mean delay for exponential service times

Assume that the service times S_u and S_d obey an exponential distribution. To optimize the performance we can apply the policy iteration method from the MDP theory [15]. This enables numerical evaluation of the gains from optimal allocation of (ϕ_u, ϕ_d) when only the number of flows in each class is known. Without any allocation constraints, the optimal policy would be a priority policy that gives priority to the class with the highest departure rate, i.e., the well-known cμ-rule [21]. By using policy iteration we can analyze how this is affected by the introduction of the allocation constraints.

The policy improvement algorithm operates as follows. Let $\mathbf{n} = (n_u, n_d)$ denote a generic state in the state space. In each state, a decision ϕ_u can be made which of the possible time allocations in Φ_u is going to be used. Furthermore, let π^i denote the policy in the i:th iteration of the algorithm, i.e., which action ϕ_u to take in each state and $\bar{n}_u(\pi^i)$ and $\bar{n}_d(\pi^i)$ correspond to the mean number of uplink and downlink flows, respectively, under the policy π^i. Also, the transition rate from state $\mathbf{n}$ to $\mathbf{m}$ when action ϕ_u is taken is given by $q_{\mathbf{nm}}(\phi_u)$. The iterated policy when minimizing the number of flows in the system is obtained from π^i by performing for each state $\mathbf{n}$ the following optimization

$$\pi^{i+1}(\mathbf{n}) = \arg\min_{\phi_u\in\Phi_u}\Big(n_u+n_d-(\bar{n}_u(\pi^i)+\bar{n}_d(\pi^i)) + \sum_{\mathbf{m}} q_{\mathbf{nm}}(\phi_u)v_{\mathbf{m}}(\pi^i)\Big), \tag{2}$$

where $v_{\mathbf{m}}(\pi^i)$ gives the so-called relative value of state $\mathbf{m}$ characterizing the difference in the expected mean number of flows for a process that is started from initial state $\mathbf{m}$ and a stationary process under the policy π^i. The relative values $v_{\mathbf{m}}(\pi^i)$ are obtained by solving the associated Howard equations,

$$n_u + n_d - (\bar{n}_u(\pi^i)+\bar{n}_d(\pi^i)) + \sum_{\mathbf{m}} q_{\mathbf{nm}}(v_{\mathbf{m}}(\pi^i) - v_{\mathbf{n}}(\pi^i)) = 0, \quad \forall \mathbf{n}.$$

Additionally, an initial policy π^0 needs to be defined, which in our numerical studies is given by the discrete D-PS policy, as described in the previous section. Thus, starting from the initial policy one first solves the associated relative values after which the first iterated policy π^1 is given by solving (2). Given the new policy one again solves the new relative values $v_{\mathbf{m}}(\pi^1)$, for all $\mathbf{m}$, after which the next iterated policy π^2 can be solved. Eventually, the iteration converges and yields the optimal policy π^*. Typically, the convergence happens after a few iterations.

The policy iteration algorithm allows us to numerically investigate how much performance can be optimized by just knowing the number of flows in each class. However, the properties of the policy iteration step (2) reveal additionally the following interesting structural result that the optimal policy is such that the optimal action in state $\mathbf{n}$ is to always choose among the corner points of the capacity region.

Proposition 1: For exponential service times, the optimal policy $\pi^*(\mathbf{n})$ in any state $\mathbf{n}$ is given by $\pi^*(\mathbf{n}) = \phi_u^{\min}$ or $\pi^*(\mathbf{n}) = \phi_u^{\max}$. Thus, the optimal policy always uses either of the extreme values in the set Φ_u.

Proof: In the policy iteration optimization step (2), for any state $\mathbf{n}$ the only terms that are affected by the choice of the action ϕ_u are the terms that correspond to a departure of either an uplink or downlink flow. Thus, the minimization corresponds to determining (the dependence on π^i has been omitted for clarity)

$$\pi^*(\mathbf{n}) = \arg\min_{\phi_u\in\Phi_u}\Big(\phi_u/S_u(v_{n_u-1,n_d} - v_{n_u,n_d}) + (1-\phi_u)/S_d(v_{n_u,n_d-1} - v_{n_u,n_d})\Big).$$

The above function is linear with respect to ϕ_u and thus the minimum value is obtained at either of the extreme values of ϕ_u, i.e., $\pi^*(\mathbf{n}) = \phi_u^{\min}$ or $\pi^*(\mathbf{n}) = \phi_u^{max}$. This holds at every iteration step of the policy iteration algorithm, which completes the proof. ■

C. SRPT-like heuristic

As shown above, for exponential service times, the optimized policy always uses either of the corner points for allocation. This is plausibly also a good principle even under general service times. On the other hand, the SRPT principle is that one always concentrates the service on those jobs that have the shortest remaining service. These observations lead us to an SRPT-like heuristic where we serve in both classes the shortest flow (in terms of remaining service time) and allocate the largest possible rate to the class with the smallest flow, in order to get rid that flow as quickly as possible under the allocation constraints.

D. Numerical results

Next we provide some illustrative numerical examples. We consider the same setting as earlier in Section III-C with symmetric arrival rates in both classes, but we assume that the service times are exponentially distributed with mean $\mathrm{E}[S_u]$ and $\mathrm{E}[S_d]$.

We first consider the overall performance in terms of the total mean number of flows in the system. Figure 4 (upper panel) shows, as a function of the load ρ, the ratio of the mean number of flows for the different policies operating in the constrained capacity region relative to the mean number of flows for the unconstrained D-PS policy. Thus, the figure depicts the loss in performance (in terms of mean number of flows) due to the allocation constraints compared with the unconstrained D-PS policy. The performance of discrete D-PS policy is never very far (at most about 20% worse) from D-PS and the policy iteration (MDP in the figure) cannot improve much the performance. The MDP results have been obtained after 5 iteration steps when starting from the discrete D-PS policy. The Balanced Fairness (BF in the figure) result shows what to expect under general service time distributions. Again, using SRPT-like size information (SRPT heuristic in the figure) can offer a significant gain, especially at moderate loads, but our heuristic may favor uplink flows too much at higher loads. This becomes clear in Figure 4 (lower panel), which shows the normalized delay ratio (recall our definition for this from Section III-C) for the different policies; indeed, the SRPT heuristic unfairness starts to increase quickly with load. However, all the other fair policies (BF and discrete D-PS) are, as expected, close to ideal in term of fairness. Interestingly, the policy iteration method also results in a very fair policy.

Finally, we make some observations on the properties of the optimal policy that can be obtained using the policy iteration algorithm. These results are actually obtained from an asymmetric arrival rate setting with $\rho_u = 0.05$ and $\rho_d = 0.85$ in order to better visualize the observations. Figure 5 (left panel) depicts the initial policy for our truncated state space (truncated at 50 flows in both classes), i.e., the allocation for ϕ_u under the discrete D-PS policy. The different colors indicate the 5 possible actions. Of special importance are the black (darkest shade) and yellow (lightest shade) areas that represent the corner points in the capacity region; the yellow area corresponds to the allocation $(\phi_u, \phi_d) = (1/10, 9/10)$ and the black area to $(\phi_u, \phi_d) = (3/5, 2/5)$. Figure 5 (right panel) presents the optimized policy after 5 steps of the policy iteration algorithm and shows that indeed only the corner points are used (as predicted by Proposition 1). Recall that without the allocation constraints, the delay optimal policy would be the cμ-rule. If one directly applies the cμ-rule to the constrained capacity region, the optimal policy would be to use either of the corner points $(\phi_u, \phi_d) = (1/10, 9/10)$ or $(\phi_u, \phi_d) = (3/5, 2/5)$, depending on the parameters. In our setting, this would correspond to $(\phi_u, \phi_d) = (3/5, 2/5)$, i.e., in Figure 5 the area would be black (except on the y-axis).

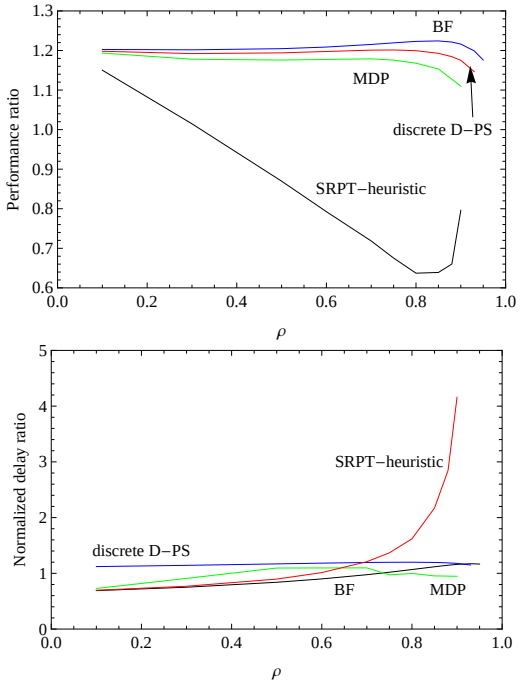

Fig. 4. The relative performance with respect to the unconstrained D-PS policy for the different policies that operate in the constrained capacity region (upper panel) and the ratio of the normalized downlink delay to the normalized uplink delay for the different policies (lower panel).

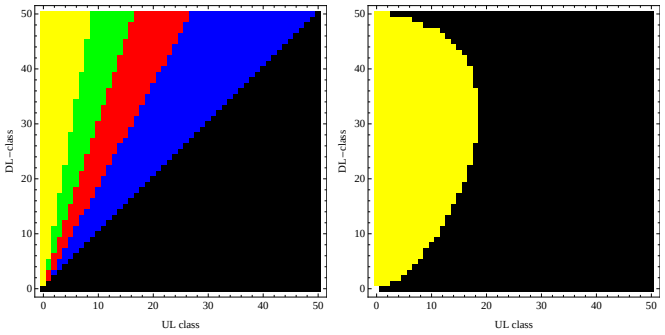

Fig. 5. Illustration of the initial policy (left panel), i.e., the discrete D-PS policy, and the optimized policy (right panel) after 5 iterations of the policy iteration algorithm. Different colors indicate the allocation among the 5 possibilities. The black and the yellow areas correspond to the corner points of the capacity region.

The shape of the area clearly suggests that the optimal policy is characterized by a switching curve with an exponential form (the turning towards y-axis is a border effect and due to the truncation of the state space) and shows how the cμ-rule is affected by the constraints in the capacity region.

V. Conclusions

We have considered the scheduling of flows (file transfers) in a single cell using dynamic TDD for controlling the resource allocation between the uplink and downlink traffic. A dynamic model was derived that takes into account stochastically arriving flows distributed randomly in space and with random sizes. The dynamic TDD resource allocation problem was modeled

as a scheduling problem in an M/G/1 queue with multiple traffic classes. The objective was to gain insights to the performance improvements achievable by dynamic adaptation of the uplink/downlink resource allocation, and to assess the impacts of practical constraints in the resource allocation.

To obtain preliminary insights, we first considered the case where the time allocation between the uplink and downlink can be done arbitrarily. We studied two fair policies (optimal static PS and dynamic PS) and two delay optimal schedulers (Gittins index and SRPT), which differ in the level of detail required from the traffic characteristics. Notably for the Gittins index policy, we established that for exponentially distributed flow sizes the policy is equivalent to a specific priority system with a continuous valued class index. Closed form expressions were provided for the performance of all policies (PS and SRPT results hold for arbitrary flow size distributions). In summary, the dynamic-PS policy represents a practical, simple and robust (recall the insensitivity property) approach showing good performance and fairness properties compared with the static-PS policy. Also, the results demonstrated that the benefits of having SRPT-like knowledge of remaining service time are considerable, but it increases unfairness between downlink/uplink.

Then we considered the impacts of practical constraints on the resource allocation between the uplink/dowlink. Motivated by the promising properties of the dynamic-PS policy, we presented the discrete dynamic-PS policy. The performance of the discrete dynamic-PS policy was analyzed by solving the steady state distribution of the corresponding Markov process for exponentially distributed service times. For general service time distributions, we applied results from Balanced Fairness, relying on the special shape of the capacity region, to provide an insensitive bound on the performance that can be used to approximate the performance of the discrete dynamic-PS policy. Also, for exponentially distributed service times we used MDP theory to establish a structural result that even in the system with allocation constraints, the optimal policy always selects among the corner points of the capacity region. The policy iteration algorithm was utilized to numerically evaluate the achievable gain from just knowing the number of jobs in the classes. Finally, based on the obtained insights, we derived an SRPT-like heuristic. The results showed that the performance of the discrete dynamic-PS policy is close to that of the unconstrained dynamic-PS, and that only detailed SRPT-like knowledge can significantly improve the performance, although, again, at the expense of increased unfairness. Thus, the discrete dynamic-PS policy is indeed a practical approach for realizing flow-level scheduling in a dynamic TDD system.

In the future research, it is, e.g., important to better address the impact of channel-aware schedulers, interference between neighboring base stations and to include correlations between flow arrivals in uplink/downlink. Also, on the theoretical side in the exponential setting, determining the form of the switching curve (even in an asymptotic sense) is an interesting problem in itself.

ACKNOWLEDGEMENT

This research has been partially supported by the HEWINETS (Dynamic Heterogeneous Wireless Access Networks) project, funded by Ericsson, Cassidian Systems and TEKES.

REFERENCES

[1] R. Susitaival, H. Wiemann, J. Östergaard, and A. Larmo, "Internet access performance in LTE TDD," in *IEEE VTC-Spring*, may 2010, pp. 1 –5.

[2] T. Bonald and A. Proutière, "Wireless downlink data channels: user performance and cell dimensioning," in *Proc. of ACM MobiCom*, Sep. 2003, pp. 339–352.

[3] H. van den Berg, R. Litjens, and J. Laverman, "HSDPA flow level performance: the impact of key system and traffic aspects," in *Proc. of ACM MSWiM*, Oct. 2004, pp. 283–292.

[4] D. C. Dimitrova, H. Van Den Berg, G. Heijenk, and R. Litjens, "Flow level performance comparison of packet scheduling schemes for umts eul," in *WWIC'08: Proceedings of the 6th international conference on Wired/wireless internet communications*, 2008, pp. 27–40.

[5] T. Bonald and H. N., "Capacity gains of some frequency reuse schemes in OFDMA networks," in *Proceedings of IEEE GLOBECOM*, 2009.

[6] C. Chiang, W. Liao, T. Liu, I. Chan, and H. Chao, "Adaptive downlink and uplink channel split ratio determination for TCP-based best effort traffic in TDD-based WiMAX networks," *IEEE Journal Selected Areas in Communications*, vol. 27, no. 2, pp. 182 –190, february 2009.

[7] M. Al-Rawi and R. Jäntti, "A dynamic TDD inter-cell interference coordination scheme for long term evolution networks," in *IEEE PIMRC*, april 2011, pp. 1 –5.

[8] T. Chahed, S.-E. Elayoubi, and E. Altman, "On design of TDD for joint uplink and downlink resource allocation in OFDMA-based WiMax," in *IEEE VTC-Fall*, sept. 2008, pp. 1 –5.

[9] J. C. Gittins, *Multi-armed bandit allocation indices*. Wiley, Chichester, 1989.

[10] S. Aalto, U. Ayesta, and R. Righter, "On the Gittins index in the M/G/1 queue," *Queueing Systems*, vol. 63, no. 1–4, pp. 437–458, 2009.

[11] ——, "Properties of the Gittins index with application to optimal scheduling," *Probability in the Engineering and Informational Sciences*, vol. 25, no. 3, pp. 269–288, 2011.

[12] L. E. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Operations Research*, vol. 16, pp. 687–690, 1968.

[13] T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo, "A queueing analysis of max-min fairness, proportional fairness and balanced fairness," *Queueing Systems*, vol. 53, pp. 65–84, June 2006.

[14] T. Bonald, S. Borst, N. Hegde, M. Jonckheere, and A. Proutiére, "Flow-level performance and capacity of wireless networks with user mobility," *Queueing Systems*, vol. 63, pp. 131–164, 2009.

[15] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[16] S. Borst, "User-level performance of channel-aware scheduling algorithms in wireless data networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 636–647, 2005.

[17] B. Sadiq and G. de Veciana, "Balancing srpt prioritization vs opportunistic gain in wireless systems with flow dynamics," in *Proceedings of 22nd International Teletraffic Congress (ITC-22)*, 2010, pp. 1–8.

[18] S. Aalto, A. Penttinen, P. Lassila, and P. Osti, "Optimal size-based opportunistic scheduler for wireless systems," *Queueing Systems*, pp. 1–26, 2012, available online.

[19] S. Aalto and P. Lassila, "Impact of size-based scheduling on flow-level performance in wireless downlink data channels," in *Proc. of ITC-20*, Jun. 2007, pp. 1096–1107.

[20] L. Kleinrock, *Queueing systems, vol II: computer applications*, 1st ed. John Wiley & Sons, 1976.

[21] C. Buyukkoc, P. Varaiya, and J. Walrand, "The $c\mu$ rule revisited," *Advances in Applied Probability*, vol. 17, pp. 237–238, 1985.

[22] L. E. Schrage and L. W. Miller, "The queue M/G/1 with the shortest remaining processing time discipline," *Operations Research*, vol. 14, pp. 670–683, 1966.

[23] T. Bonald and J. Virtamo, "Calculating the flow level performance of balanced fairness in tree networks," *Performance Evaluation*, vol. 58, pp. 1–14, October 2004.