Invited Paper: Context-Aware Schedulers: Realizing Quality of Service/Experience Trade-offs for Heterogeneous Traffic Mixes

Arjun Anand and Gustavo de Veciana
Department of Electrical and Computer Engineering
The University of Texas at Austin

***Abstract*—Modern broadband wireless networks support application mixes, with different, possibly complex, application/user Quality of Service/Experience (QoS/QoE) metrics. The central problem underlying resource allocation for such systems is realizing QoS/QoE trade-offs given the dynamic loads and capacity variability they would typically see. The paper explores a framework for context-aware scheduling based on: (1) context-aware flow classification and management, and (2), complementary base station scheduler. Motivated by typical flow-size distributions for current traffic and characteristics of the associated delay optimal (Gittins index-based) schedulers we propose a novel flow and channel-aware scheduler which meets our design objectives. Using a combination of analysis and simulation we explore the achieved QoS/QoE trade-offs across a dynamic mix of traffic, in particular: 1) mobile web browsing and small file delays; 2) stored streaming video quality vs re-buffering; 3) throughput of larger file downloads. They suggest improved QoS/QoE trade-offs vs traditional proportionally fair schedulers and are robust to the network load.**

I. Introduction

Modern mobile broadband networks support traffic associated with a highly diverse and evolving set of applications including: stored media streaming, web browsing, file transfers ranging from small (e.g., instant messages, email) to possibly large (e.g., images, movies, software updates), along with real-time media (e.g., live sports) and machine-to-machine data exchanges. This makes the design of resource allocation policies and base station schedulers a particularly hard problem.

There are three interrelated challenges to addressing this problem. First, the impact of resource allocation on an application's Quality of Service (QoS) or user's Quality of Experience (QoE) can be quite different, and in some cases may even be hard to characterize altogether, e.g., video QoE. Hence, we have to look at scheduler designs which use different performance metrics for different types of applications.Therefore, the schedulers have to be *context-aware*, i.e., either explicitly or implicitly aware of the nature of applications and/or relationship between resource allocation and user's QoE.

Second, wireless systems are subject to substantial temporal variability and spatial heterogeneity in capacity. Even for stationary users, wireless channel capacity can fluctuate by several orders of magnitude from the cell's 'center' to its 'edge.' Further, the number of active users can change dramatically as they join, move and leave, and the overall network loads and traffic mixes can vary throughout the day. Therefore, the allocation of resources should be suitable for such dynamic settings.

The third challenge is managing trade-offs amongst heterogeneous traffic mixes, particularly when the network becomes congested – i.e., how to optimize a *graceful degradation* in QoS/QoE when resources become scarce. Although congested resources are often the focus of design one must also consider how to optimize resource allocation when the system has light/moderate loads. This challenge associated with realizing trade-offs is really the crux of the problem underlying scheduler design and yet is poorly understood and poorly reflected in state-of-the-art schedulers. Let us illustrate this via several examples:

1) Web browsing vs large file downloads. Web browsing sessions involve human interaction on the order of seconds, so the QoE metric of interest is maintaining responsiveness, i.e., delays on the order of seconds to download the typically small files associated with web content for mobile devices. By contrast, large files take a long time, so one might posit the relevant QoS metric is long term throughput. Clearly a scheduler that prioritizes small files associated with web browsing and other applications, over large files achieves a good QoS/QoE trade-off for the mix.

2) Video QoE management at congested base stations. Modern stored video streaming protocols, such DASH (Dynamic Adaptive Streaming over HTTP), are rate adaptive, i.e., they adapt the video rates, and associated quality, to network congestion an/or the risk of playback re-buffering. Consider a setting where a base station serves users with heterogeneous capacity (center/edge users) via a proportionally fair scheduler, i.e., allocations which are directly proportional to the user's capacity. For light to moderate base station loads edge users might see reduced video quality vs those at the cell center, which is reasonable. Under high loads, however, edge users will start to see playback re-buffering, i.e., QoE which is unacceptable. Thus for congested resources the scheduler should be more aggressive in shifting resources from cell center to edge users.

The above exemplify some of the complex trade-offs

This work is supported by Futurewei Technologies

978-1-5090-1311-1/16/$31.00 ©2016 IEEE

	Stored Video Streaming		Web browsing/Small files	Large files
Network load	Video quality	Re-buffering	Mean flow delay	Mean throughput
Low	High	Low	Low	High
Medium	Medium	Low	Low	High
High	Low	Low	Low/Medium	Medium

TABLE I
SCHEDULER DESIGN OBJECTIVES: QOS/QOE TRADE-OFFS ACROSS APPLICATIONS VS NETWORK LOADS.

base station schedulers need to make across heterogeneous applications. Realizing such trade-offs through the design and analysis of *context-aware* schedulers is the focus of this paper. Table I exhibits an example of the high-level goals we aim to achieve for a mix of stored video streaming, web browsing, and file transfers. We shall focus on the following natural QoS/QoE metrics which represent a simplification of the more complex models discussed further in the related work.

1) Mean delay for small flows. Most small flows are currently due to web traffic, for which the overall transfer delay (time to display) is the key goal. It is of interest to limit such delays to less than a second, in order to maintain interactivity. Further, ideally these delays should not be too sensitive to other network loads, e.g., video streaming, large files etc.

2) Video quality and re-buffering for stored video streaming. The first priority is to avoid client re-buffering, beyond this one would like to achieve good average video quality depending on the load and the users' channel condition.

3) Throughput for large files. It is reasonable for large files to see delays proportional to their size. Therefore, one would expect the perceived throughput to be the relevant metric, though it might be affected by the overall system load and mix of traffic.

Before we discuss our work in more detail, let us put it into proper context based on the substantial previous work considering base station scheduling from different perspectives.

A. Related work.

Modeling QoS/QoE. Traditional QoS metrics such as throughput, packet delays and jitter, may not properly reflect user experience. For this reason there has been significant interest in better modeling user perceived QoE for various applications. For example, for interactive web browsing, QoE was found to be well modeled as a function of the delay of transactions, see [1], [2]. In particular [1] shows that web browsing QoE is an S-shaped function of transaction delay, whereas [2], propose polynomial functions of transaction delays. These, and other recent efforts reinforce the need to look at QoE metrics depending on flow (transaction) delays. Perhaps the simple lesson learned here is that one would like to see small transaction delays, below some level, but further reductions do not have a high marginal benefit. We shall embrace this principle in this paper. Similarly, there has been substantial recent interest in modeling streaming video QoE including aspects of the quality of the reproduced video, possibly quality variability, re-buffering, and start up delays, see [3] and references therein. In general there is agreement that avoiding re-buffering is the priority if one is to improve user perceived QoE, see [4].

Scheduling. Traditional work focused on scheduling for elastic traffic[1] focused on 'fair' rate allocation by using utility maximization approaches in the full buffer model, see e.g., [5]–[7] for detailed surveys. In general this fails to directly account for the dynamic nature of traffic and indeed the flow-level delays which may be most related to user perceived QoE.

There is also substantial work on queue-based schedulers addressing stability and/or QoS for real-time traffic, e.g, VoIP in LTE networks. Most of this work augments the utility-based schedulers such as proportionally fair (PF) with the current queue lengths of users, see e.g., [7]. A weakness of this work remains the lack of focus on flow level metrics and ability to multiplex and control performance when there are user dynamics.

Another area of substantial research is network scheduling and transport for modern DASH-like video streaming, see e.g, [8]–[10]. In general these works strive to optimize the video client behaviour as well BS/core network scheduling to optimize video QoE with constraints on re-buffering time, or fraction of time low quality video is delivered. These works do not fully address the impact of flow level dynamics and in particular the sharing of resources by heterogeneous applications. Still in the sequel we shall adopt [8] as a representative mechanism to assess our context-aware scheduler.

Finally, there has been some work on scheduling to address flow-level delays which draws from a rich body of work in queuing theory, see e.g., [11]–[18]. These works address the minimization of average flow delay for traffic having a mix of small and large flows, i.e., the so called mice and elephants. It is well known that if a scheduler knows the required processing time of flows, the Shortest Remaining Processing Time policy minimizes the mean delay, see e.g., [17]. If such information is not available, the scheduler may infer this based on cumulative service to date and/or use prior knowledge of the flow size distribution. This is represented by schedulers such as the Foreground-Background (FB) or Least Attained Service (LAS), Multi-Level processor sharing, FCFS + FB, etc, which are delay optimal in various settings depending on the flow-size distribution, see e.g. [12]–[14], [17]. These above works for the most part do not address wireless networks

[1]Traditionally interactive web browsing, large file downloads, emails etc are classified into a single category called best effort elastic traffic.

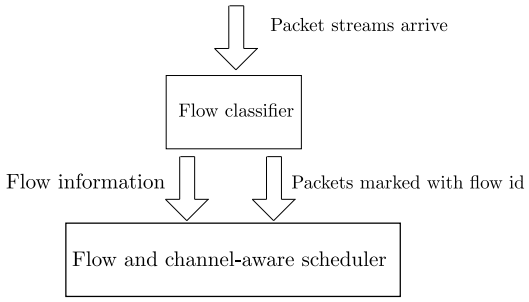

Fig. 1. The block diagram for our context-aware scheduler.

where different flows may see heterogeneous and/or changing wireless capacity. Exceptions include downlink scheduling studied in [11], [19]. We will draw on this previous theoretical work in developing our own approach and in our effort to tackle QoS/QoE trade-offs across heterogeneous traffic.

B. Our Contributions

In this paper we recognize that for many applications the QoS/QoE is tied to flow-level performance. For web browsing sessions, flows are associated with web pages that are being downloaded. Similarly modern stored video streaming can be viewed as a stream of 'flows' associated with video segments whose quality and thus size may be adapted to network congestion. Thus the QoE for video depends on the delays/arrivals of the stream of flows. We propose a two-level framework for context-aware scheduling. The upper block, called the *flow classifier*, realizes context-aware decisions, regarding applications flows and possible trade-offs e.g, managing re-buffering amongst video streams. The lower block, implements a flow- and channel-aware scheduling algorithm, aimed at reducing delays for small flows without requiring prior knowledge of their size. To that end we study the characteristics of mean delay optimal Gittins index scheduler for an idealized model of a wireless BS serving users with heterogeneous capacity and for a class of flow size distribution typical of today's networks. Extensive simulations are used to compare our context-aware scheduler to traditional proportional fair schedulers. In particular we show that our approach is able to achieve the desired trade-offs (see Table I) in QoS/QoE amongst streaming video, web browsing and large file transfers and do so robustly over a range of network loads.

C. Organization

The paper is organized as follows. In Sec. II we present an architecture of our context-aware scheduler. Its design and analysis are discussed in Sec. III. A performance analysis via simulation is presented in Sec. IV, followed by our conclusions in Sec. V.

II. Context-Aware scheduler

Our context-aware scheduler consists of two modules, namely, a flow classifier and a flow and channel-aware scheduler. The block diagram is shown in Figure 1. We describe the two blocks in detail.

A. Flow classifier

Packet streams arrive to the flow classifier block which realizes context-aware decisions. This block may be implemented at the BS itself or in the core network. Its main functions are twofold.

1) Manage flow information. It distinguishes flows based on their application type and marks the packets associated with a flow with a unique flow id. This information is later used by the scheduler block. It may decide when a flow has completed based on a threshold on the gap in inter-packet arrivals. The flow classifier exchanges control signals and flow level information with the scheduler, for example, to signal the initiation of a new flow. It may also gather meta-data associated with the flows which may be shared with the scheduler, e.g., knowledge of video segment playback duration.

2) Monitor performance. We envisage a flow-classifier that may actively manage performance, e.g., video QoE. For example, it may aim to ensure sustained playback for video clients without re-buffering. To that end, it may monitor video streams to ensure they are not starved of resources by the scheduler block. Suppose that all video users are continuously watching the video. Otherwise, the video clients stop requesting new segments and our flow classifier detects that streaming has completed based on inter-packet intervals. Consider the following simple strategy to prevent re-buffering. The flow classifier samples the *deficit* of video streams whenever a flow completes service.[2] Let $\mathcal{N}$ be the set of video streams in the system. Let τ_i, $i = 1, 2, \ldots$ be the instants at which flows, i.e., video segments are delivered. If $s_i\left(t_1, t_2\right]$ is the total number of segments downloaded by video stream i between time t_1 and t_2, then the deficit for the i^{th} stream $d_i\left(\tau_k\right)$ is defined as

$$d_i\left(\tau_k\right) := \max\left\{d_i\left(\tau_{k-1}\right) + \tau_k - \tau_{k-1} - \tau_{\text{seg}}\ s_i\left(\tau_{k-1}, \tau_k\right], \overline{\gamma}\right\}, \quad (1)$$

where τ_{seg} is the video playback duration of a segment and $\overline{\gamma} \leq 0$ is a suitably chosen threshold. A positive deficit at any time means that the number of segments downloaded until then is not sufficient for sustained playback, and the video client is in re-buffering state. A negative $\overline{\gamma}$ puts a more stringent constraint on re-buffering. Let $\mathcal{D}(\tau_k)$ be the set of flows for which the deficit is strictly greater than $\overline{\gamma}$ at time τ_k. If $\mathcal{D}(\tau_k)$ is non-empty, then the flow classifier block disables the set of flows $\mathcal{N} \setminus \mathcal{D}(\tau_k)$ till τ_{k+1}, i.e., the flows in the set $\mathcal{N} \setminus \mathcal{D}(\tau_k)$ do not contend for the radio resources in the next $\tau_{k+1} - \tau_k$ seconds. This ensures that the deficient video streams are given priority over the streams which have a sufficient number of segments in the playback buffer.

B. Flow and channel-aware scheduler

This block allocates the radio resources to flows. The scheduling policy specifies which flows are to be served at each slot. It may use the current Channel Quality Indicator

[2]Video segments are marked as flows by flow classifier.

metric (CQI) of users with active flows and/or the flow state information, e.g., the amount of service given to a flow. We discuss its design and analysis in the next section.

III. Design and analysis of flow and channel-aware scheduler

A. Idealized queuing model

To motivate our flow and channel-aware scheduler we shall revisit an idealized queuing model based on a traditional multi-class M/GI/1 queue.

Arrival process. We shall model the arrival process of flows to the system as a Poisson process of appropriate rate. Each flow is associated with a user having possibly different channel strengths and thus associated peak (average) transmission rates. We classify flows into K distinct classes based on their current transmission rates. The rate of arrival for each class is given by λ_i, $i = 1, 2, \ldots, K$. Let c_i be the transmission rate for the i^{th} class and let $c_1 < c_2 \ldots, c_K$. We assume, for now, that a flow's transmission rate remains fixed throughout its lifetime. However, class changes can be easily incorporated into our scheduling algorithm – this is addressed in Sec. IV.

Flow size distribution. Our scheduler sees a heterogeneous mix of flows associated with interactive web traffic and small to large file downloads. Therefore, from a statistical point of view, the scheduler sees a concentration short and medium sized flows and few large flows.[3] This is very well captured by the NBUE + DHZ (β) flow size distributions introduced below.

Let X denote the random variable (r.v.) modeling the size of a typical flow. Let $G_X(x)$, $g_X(x)$, and $\overline{G}_X(x)$ be the cumulative distribution function (c.d.f.), probability density function (p.d.f.), and complementary c.d.f. (c.c.d.f.), respectively. We assume that the c.d.f. is a continuous function of the flow size. Define the hazard rate function by $h_X(x) := \frac{g_X(x)}{\overline{G}_X(x)}$. A distribution is said to be of type NBUE + DHZ (β) if:

1) When the flow size is less than β bits, then the distribution is of the type New Better Than Used in Expectation (NBUE), i.e., the expected residual size of a flow which has attained service less than β bits is less than the original expected size of the flow. This implies that $\forall a \leq \beta$,
$$\mathbb{E}[X] \geq \mathbb{E}[X - a | X > a]. \tag{2}$$
2) When the flow size is more than β bits, then the flow size distribution has Decreasing Hazard Rate (DHZ). This means that $h_X(x)$ is decreasing function of x for $x > \beta$. The DHZ property is a sufficient condition for a distribution to have an increasing mean residual file size.

An example of a distribution which is NBUE + DHZ (β) is the *Exp. + Pareto* distribution which is given below:

$$\overline{G}_X(x) = \begin{cases} \exp(-\mu x), & x < \beta, \\ \exp(-\mu\beta)\left(\frac{\beta}{x}\right)^{\alpha}, & x \geq \beta, \end{cases} \tag{3}$$

[3]See extended version of paper for statistical analysis and fit of data collected by [20] showing this is indeed the case.

where $\mu > 0$ and $\alpha > 1$, where α models the tail of the Pareto distribtion of large files. More examples are given in [12].

Our preliminary exploration of measured data in [20] shows that flow size distributions on today's networks are very well modeled by NBUE + DHZ (β) distributions with Pareto tail. Due to space constraints, we have included it in the extended version [21]. Therefore, in this paper we mainly consider distributions with Pareto tail and we call them NBUE + Pareto (α, β).

B. Mean delay optimal policy

When flow sizes are not directly available, the Gittins index scheduling policy is known to minimize the expected delay in an M/GI/1 queue [16]. Below we shall introduce the Gittins index and discuss some of its important properties derived in [12], [13]. We shall build on these properties to derive the optimal scheduling policy for our multi-class wireless setting.

Gittins Index. We shall first study a single class M/GI/1 queue which serves flows at unit rate. This means that a flow of size x bits will take x seconds to complete service. Consider a flow which has already been received a bits of service. Define $J(a, \Delta)$ for $\Delta \geq 0$ as

$$J(a, \Delta) := \frac{R(a)}{C(a)}, \tag{4}$$

where $R(a) = \left(\overline{G}_X(a) - \overline{G}_X(a + \Delta)\right) / \overline{G}_X(a)$ and $C(a) = \left(\int_0^{\Delta} \overline{G}_X(a + t)\, dt\right) / \overline{G}_X(a)$. Here $R(a)$ and $C(a)$ are the probability that a flow which has attained service of a bits will complete and the expected additional time required by the flow to complete when it is allocated Δ seconds of service, respectively. Therefore, $J(a, \Delta)$ is the ratio of expected *reward* to the expected *cost* of giving a Δ seconds of service to a flow that has already received a bits of service. The Gittins index defined in [22] is given by

$$\mathcal{G}_X(a) = \sup_{\Delta \geq 0} J(a, \Delta). \tag{5}$$

There may be many values of Δ that maximize the above expression with a possible value of $+\infty$. We define $\Delta^*(a)$ as

$$\Delta^*(a) = \sup_{\Delta \geq 0} \{\Delta : J(a, \Delta) = \mathcal{G}_X(a)\}. \tag{6}$$

A scheduler is which serves the flow with the highest Gittins index at all times is called the *Gittins index scheduler*.

We summarize the important properties of the Gittins index for NBUE + DHZ (β) type distributions, these are derived in [12], [13].

Proposition 3.1: The Gittins index $\mathcal{G}_X(\cdot)$ for NBUE + DHZ (β) distributions has the following property:

(a) $\Delta^*(0) \geq \beta$.
(b) For all $a < \Delta^*(0)$, $\mathcal{G}_X(a) \geq \mathcal{G}_X(0)$.
(c) For all $a \geq \beta$, $\mathcal{G}_X(a)$ is decreasing and $\mathcal{G}_X(a) = h_X(a)$.
(d) If $h_X(x)$ is continuous and $0 < \Delta^*(0) < \infty$, then $\mathcal{G}_X(0) = \mathcal{G}_X(\Delta^*(0)) = h_X(\Delta^*(0))$.

Comments. (a) and (b) above imply that if a flow which has not received any prior service is selected for service, it would receive $\Delta^*(0) \geq \beta$ seconds of server time. Once it begins service, other flows in the system which have not received any service previously would not preempt it. Property (c) implies that the Gittins index is a decreasing function of x, for $x > \beta$. This is because of the DHZ tail which makes it less beneficial for the system to serve large flows.

Next we discuss the Gittins index scheduler for our wireless BS model based on multi-class class M/GI/1 queue.

C. Optimal scheduler for multi-class M/GI/1 queuing system

Consider the multi-class M/GI/1 queuing model for the BS. A flow of size x bits in i^{th} class requires x/c_i seconds of server time. Therefore, the mean service time associated with a flow in i^{th} class is $\mathbb{E}[X]/c_i$. For now we shall assume that at any time $t \geq 0$ only one flow is scheduled for transmission using the entire bandwidth available.

Before we derive the optimal Gittins index scheduler for this model, we consider the Gittins index for our multi-class system. The Gittins index in this setting depends on both the class of a flow and its attained service. This is because when the server allocates Δ seconds of service time to a flow, the probability that it completes service within the Δ seconds and the expected time it takes to complete service depend on the transmission rate of its class. We shall express the Gittins index of a flow in i^{th} class, $\mathcal{G}_i(\cdot)$, in terms of the Gittins index $\mathcal{G}_X(\cdot)$ associated with an M/GI/1 system where flows are served at unit rate.

Lemma 3.2: Suppose a flow of class i has attained x bits of service, then its Gittins index $\mathcal{G}_i(\cdot)$ is given by:

$$\mathcal{G}_i(x) = c_i \mathcal{G}_X(x). \tag{7}$$

Proof: Proof is given in the extended version [21]. ■

The Gittins index scheduler requires exact knowledge of the index as a function of the service given to a flow. Thus in order to compute the Gittins index we require the knowledge of the distribution of flow sizes. This information may not available in practice. Therefore, we require a robust approximation to the Gittins index scheduler which is based on easily measurable statistical properties such as the mean flow size. In the sequel we discuss some of the key characteristics of the Gittins index scheduler which will be used to motivate our scheduler design.

D. Qualitative characteristics of the optimal scheduler

Fig. 2 illustrates all the properties of the Gittins index mentioned in Prop. 3.1 and in Lemma 3.2. At any given time, the states of flows present in the system can be visualized as points on the Gittins index curves based on the service they have attained. The x-axis of a point represents the number of bits served for that flow, and y-axis is its Gittins index based on its class, for example, a new flow arriving to class i is represented by the point $(0, \mathcal{G}_i(0))$. As the flows get served they move along the Gittins index curve.

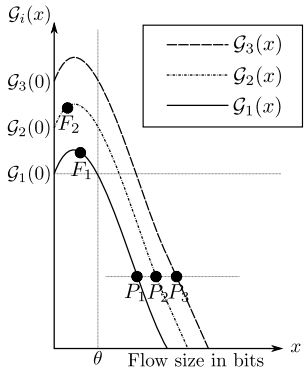

Fig. 2. Illustration of Gittins index curves as function of the flow size for a multi-class M/GI/1 queuing system.

Define $\theta := \Delta^*(0)$. We call θ as the *cross-over threshold.* Later in this section, we will see that the Gittins index policy treats flows that have received less than θ bits of service and more than θ bits of service differently. We shall use this property to develop our approximation to the Gittins index scheduler.

Consider the characteristics of the optimal scheduling policy when all the flows in the system have received less than θ bits of service. The flow which is in state F_1 on the Gittins index curve in Fig. 2 has received less than θ bits of service. Its Gittins index is greater than $\mathcal{G}_1(0)$. This means it enjoys a higher priority over new arrivals to class 1 and over the flows in Class 1 which have not been served till now. Therefore, the scheduling policy is First Come First Serve (FCFS) among the Class 1 flows which have received service less than θ bits. This is true for other classes too. This FCFS policy is a result of the NBUE property of the flow size distribution when flow sizes are less than β bits (which is less than θ).

In scenarios where the capacities of various classes are widely separated, if $i > j$, then $\mathcal{G}_i(x_i) > \mathcal{G}_j(x_j), \forall x_i, x_j \leq \theta$. Therefore, among the flows which have attained service less than θ bits, flows with higher transmission rates should preempt the flows with lower transmission rates. For example, a flow in state F_2 should preempt a flow at F_1. This implies that the scheduling policy is *multi-class preemptive FCFS* for all flows which have attained service less than θ bits i.e., the policy is FCFS for flows within a class and flows in classes with higher transmission rates can pre-empt flows with lower transmission rates.

Next we discuss the characteristics of the optimal scheduling policy for long flows which have received a large amount of service. Consider points P_1, P_2, and P_3 on the Gittins index curves in Fig. 2. They all have the same value for their Gittins index. Let M be the total number of flows in these states. If we consider distributions with Pareto tails, i.e., the tail probability decays as $1/x^\alpha$, $\alpha > 1$, then it is clear that the Gittins index scheduler serves these M flows according to the Processor Sharing (PS) discipline with equal fraction of time given to all the flows, see [11]. Since each flow receives an equal

fraction of time, they get rates proportional to their channel capacities, i.e., rate allocation is Proportionally Fair (PF).

Another key observation is that all really long flows in the system which already received a large amount of service have a lower Gittins index than new arrivals and the flows which have received service less than θ bits. This is due to the DHZ property of the tail.

To summarize, the optimal scheduler has the following characteristics

1) All flows which have received cumulative service less than θ bits are served based on pre-emptive priority for classes with higher c_i and FCFS within classes.
2) Flows which have received a large cumulative service are eventually served using PF scheduling.
3) Flows which have received service less than θ bits have priority over those which have already seen a large cumulative service.

The above characteristics motivate an approximation to the optimal Gittins index scheduler. This is explained next.

Let f be an active flow. Its time of arrival is denoted by $f.t$. At any point in time flows in a given class i are partitioned into two sets: $\mathcal{L}_i$ denoting those that have received less than or equal to θ bits and $\mathcal{H}_i$ the remaining flows. Define $\mathcal{L} := \cup_{i=1}^{K}\mathcal{L}_i$ and $\mathcal{H} := \cup_{i=1}^{K}\mathcal{H}_i$. The sets $\mathcal{L}$ and $\mathcal{H}$ include all active flows which have received less than θ bits of service and more than θ bits of service, respectively. If $\mathcal{A}$ and $\mathcal{B}$ are two sets, then $\mathcal{A} \succ \mathcal{B}$ implies that the flows of $\mathcal{A}$ are given preemptive priority over the flows of $\mathcal{B}$. Next we introduce our approximation to Gittins index scheduler which we will call p-FCFS + PF (θ).

E. Approximation to Optimal Scheduling – p-FCFS + PF (θ)

To specify a scheduling policy, we need to specify how flows are prioritized among the sets $\{\mathcal{L}_i\}_{i=1}^{K}$ and $\{\mathcal{H}_i\}_{i=1}^{K}$. Once we decide the priority between sets, we specify how resources are allocated to flows within these sets. We shall give priority to various sets in the following manner – $\mathcal{L}_K \succ \mathcal{L}_{K-1} \ldots \succ \mathcal{L}_2 \succ \mathcal{L}_1 \succ \mathcal{H}$. In $\mathcal{L}_i$, the flow which has the earliest arrival time has the highest priority. In $\mathcal{H}$, all flows have the same priority. Thus in each slot we propose to implement Algorithm 1.

This is a simple low complexity scheduling policy which approximates the optimal Gittins index scheduler for small and really large flows. It only requires knowledge of one parameter– the cross-over threshold θ. Informally the cross-over threshold distinguishes between small and large flows. Choosing the right value of θ is critical to the mean delay performance of the scheduler. If contextual information about various types of traffic is available to the network operator, then θ can be chosen such that the delay critical applications are given priority. For example if we know the maximum size of video segments, then for good performance of video streams, one can choose a value of θ which is slightly more than the maximum video segment size. If contextual information is not available and we only have knowledge of the distribution of the flow sizes of the traffic mix, then we have an analytical characterization of θ based on its definition. One can develop an approximate expression for θ which depends on two measurable properties – the mean flow size and the exponent of decay of the tail probability of flow size distribution.

Algorithm 1 p-FCFS + PF (θ)

$\{\mathcal{L}_i, \mathcal{H}_i\} \leftarrow$ FLOW MANAGEMENT(θ)
if $\mathcal{L} \neq \phi$ **then**
  $i^* = \text{argmax}_i \{i|\mathcal{L}_i \neq \phi\}$
  Serve flow $f^* = \text{argmin}_f \{f.t|f \in \mathcal{L}_{i^*}\}$
else
  if $\mathcal{H} \neq \phi$ **then**
    Serve all flows in $\mathcal{H}$ according to PF scheduling policy.
  end if
end if
procedure FLOW MANAGEMENT(θ)
  Update each $\mathcal{L}_i$ with new arrivals.
  Move flows with attained more than θ bit of service from the corresponding $\mathcal{L}_i$ to $\mathcal{H}_i$.
  Remove flows that have completed service.
end procedure

Proposition 3.3: For NBUE + Pareto (α, β) distribution, θ is obtained by solving the following fixed point equation:

$$\theta = \alpha \left[\frac{\mathbb{E}[X] - P(X > \theta) \frac{\alpha\theta}{\alpha - 1}}{P(X \leq \theta)} \right], \tag{8}$$

where X is the random variable denoting the flow size. For large enough values of α, $\theta \approx \alpha\mathbb{E}[X]$.

Proof: Proof is given in the extended version [21]. ■

For $\alpha > 2$, our approximation is quite close to θ. Detailed comparisons between θ and its approximation are given in [21].

IV. PERFORMANCE EVALUATION

In this section we present the simulation results for our proposed approach. We compare its performance with that of the PF scheduler. We consider a single BS serving 9 video streaming users and a dynamic number of active web browsing sessions and file downloads. The BS uses slotted time with slot duration $\tau_{\text{slot}} = 0.01$ sec. It makes scheduling decisions at the beginning of each slot. The users are located at varying distances from the base station and therefore, have heterogeneous channel strengths. The channel variations due to mobility are modeled by Markov Chain. The marginal distribution of this Markov chain is same as appropriately scaled versions of the channel strength distribution obtained from an HSDPA system. See [8] for more details on the generation of channel realizations. We classify the users into 10 different classes based on their channel strengths in each slot. Due to the time varying nature of wireless channels, the users may move from one class to another.

The flow sizes of the mix of web browsing and file downloads are modeled as a Pareto distribution with the parameters

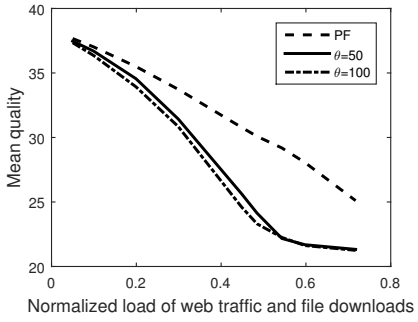

Fig. 3. Mean quality vs normalized load of web traffic and file downloads.

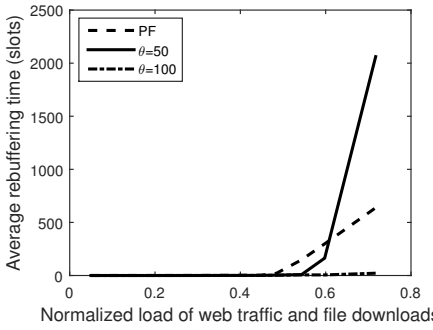

Fig. 4. Mean re-buffering time vs normalized load of web traffic and file downloads.

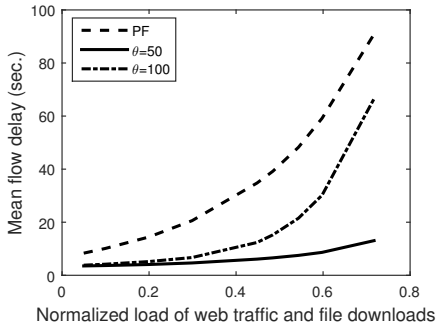

Fig. 5. Mean delay for flows less than 100 Kbits vs normalized load of web traffic and file downloads.

$\beta = 40$ Kbits and $\alpha = 5$. These flows arrive to the system as a Poisson process with suitable rate, independent of the video traffic in the system. We classify the flows with size less than 100 Kbits size as interactive web traffic and the rest as file downloads.

The stored video delivery model which we simulate mimics the DASH framework. Similar simulation model for video has also been studied in [8]. The video users view different parts of three open source movies, namely, Oceania, Route 66, and Valkama. The video segments sent are of one second playback duration. Each video segment has 6 different representations of varying quality and segment sizes. The sizes of various representations in the increasing order of quality are 100, 200, 300, 500, 900, and 1500 Kbits/segment. We use MSSSIM-Y metric (see [23]) for video segments to measure the mean quality of the video stream delivered.

The video client application with each user is such that it requests the next video segment only after the previous segment is delivered. The video client can buffer at most ten video segments. When the buffer is not full the client requests the next segment using the state-of-the-art algorithm QNOVA proposed in [8]. QNOVA is a client application which takes into account mean-variability trade-offs in quality, pricing constraints and re-buffering constraints to request appropriate representation for next video segment. In our simulation we adjust QNOVA such that it does not consider variability in quality across video segments nor pricing constraints. We also relax the re-buffering constraints in QNOVA because our context-aware scheduler takes care of the re-buffering events.

Figures 3 and 4 plot the mean quality of video streams and the average re-buffering time as a function of the normalized load of web traffic and file downloads, respectively. Normalized load is defined as the total data rate of web traffic and file downloads arriving to the system divided by the mean transmission rate for flows. It is a proxy for the fraction of system utilization by web traffic and file downloads. Figures 5 and 6 plot the mean flow delay for interactive web traffic and the mean throughput for file downloads as a function of its normalized load. We compare our context-aware scheduler with the PF scheduler which does not use contextual information. Through simulations we found that θ between 50 Kbits and 100 Kbits give good results. The key results are:

1) Trade-off between mean quality and mean delay at lower loads. In Fig. 5, we observe that our scheduler improved the mean delay for interactive web traffic by atleast 54% for loads less 0.4, when compared to the PF scheduler. This is because it expedites flows of size less than θ via the flow and channel-aware scheduler block in our context-aware scheduler. Thus there is slight reduction in the mean video quality for

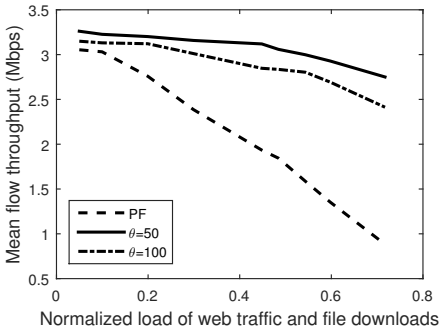

Fig. 6. Mean throughput for flows greater than 100 Kbits vs normalized load of the interfering web traffic and file downloads.

system loads less 0.4. Since the lowest quality representation of video is 100 Kbits, $\theta = 50$ Kbits selectively expedites short flows over video and file downloads much more than $\theta = 100$ Kbits. Therefore, $\theta = 50$ Kbits gives better mean delay performance.

2) Robustness to loads. In Fig. 5, we observe that for our scheduler the mean delays for flows less than 100 Kbits size do not vary much for loads less than 0.4. For example, when $\theta = 50$, the mean flow delay increases by 85% when the load increases from 0.1 to 0.4. However, for the PF scheduler, the mean delay increases by 323% for the same range of loads. This robustness is a result of our scheduler favoring short flows and forcing the video clients to request lower representations as the load increases. Therefore, the video streams adapt better to the changing system loads versus PF scheduler.

3) Trade-off between mean quality and re-buffering at higher loads. Figure 4 shows that our scheduler accommodates a much higher load of interfering web traffic and file downloads without resulting in re-buffering. For $\theta = 50$ Kbits, our context-aware scheduler can sustain video playback without re-buffering till a load of 0.55. This is 15% gain over PF scheduler which has non-zero re-buffering time at a load of 0.48. Similarly for $\theta = 100$ Kbits we see a gain of 46%. For $\theta = 100$ Kbits the gain is higher because we give priority to all flows less than 100 Kbits, which include the lowest quality video segments. The price we pay for avoiding re-buffering is the reduction in mean quality at higher loads, say between 0.4 to 0.6. There are two reasons for this reduction in mean quality. First, our scheduler favors flows of size less than θ. Second, when the system is congested, the re-buffering avoidance mechanism in the BS prevents users which have sufficient segments in their playback buffers from obtaining the radio resources.

4) Increased throughput. Figure 6 shows that our schedulers have a higher mean throughput for flows of size exceeding 100 Kbits. For a load of 0.4, our scheduler has atleast a gain of 46%. Our flow and context-aware scheduler significantly reduces the delay for flows of size slightly larger than θ Kbits. This results in the increased throughput for our scheduler. However, we note that for really large flows the mean throughput in our scheduler could be less than that of PF scheduler, but such events occur very rarely.

V. Conclusions

In this paper, we aimed to design and study scheduler achieving robust QoS/QoE trade-offs amongst heterogeneous applications/users sharing a Base Station. Robustness here corresponds in part to the possibility of changing the nature of the trade-offs as the network loads increase so as to better address the sensitivity of various applications/users to congestion. Through a combination of analysis and extensive simulations we have evaluated our proposed framework and believe that it has met the objectives we set for mixes of streaming video, web browsing, and file transfers which are the lions share of today's wireless data traffic.

References

[1] M. Proebster, "Improving the quality of experience with size-based and opportunistic scheduling," in *Proc. Int. Symp. on Wireless Commn. Sys. (ISWCS)*, Aug. 2014, pp. 443–448.

[2] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz, P. Mogensen, and J. M. Lopez-Soler, "QoE oriented cross-layer des. of a resource alloc. alg. in beyond 3G systems," *Computer Commn.*, vol. 33, no. 5, pp. 571–582, Mar. 2010.

[3] K. Seshadrinathan and A. Bovik, "Automatic prediction of perceptual quality of multimedia signalsa survey," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 163–186, 2011.

[4] R. Mok, E. Chan, and R. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. IFIP/IEEE Int. Symp. on Integrated Netw. Management*, May 2011, pp. 485–492.

[5] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," *IEEE Commn. Surveys Tutorial*, vol. 15, no. 4, pp. 1671–1688, Jan. 2013.

[6] M. Andrews, "A survey of scheduling theory in wireless data networks," in *Wireless Commun.*, ser. The IMA Volumes in Mathematics and its Applications. Springer New York, 2007, vol. 143, pp. 1–17.

[7] B. Sadiq, R. Madan, and A. Sampath, "Downlink scheduling for multiclass traffic in LTE," *EURASIP J. Wirel. Commun. Netw.*, vol. 2009, pp. 14:9–14:9, Mar. 2009.

[8] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of dash-based video delivery in networks," in *Proc. INFOCOM*, Apr. 2014, pp. 82–90.

[9] H. Kowshik, P. Dutta, M. Chetlur, and S. Kalyanaraman, "A quantitative framework for guaranteeing QoE of video delivery over wireless," in *Proc. INFOCOM*, Apr. 2013, pp. 290–294.

[10] D. Bethanabhotla, G. Caire, and M. Neely, "Utility optimal scheduling and admission control for adaptive video streaming in small cell networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 1944–1948.

[11] S. Aalto and P. Lassila, "Impact of size-based scheduling on flow level performance in wireless downlink data channels," in *Managing Traffic Performance in Converged Netw.*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4516, pp. 1096–1107.

[12] S. Aalto and U. Ayesta, "Optimal scheduling of jobs with a DHR tail in the M/G/1 queue," in *Proc. of the 3rd Int. Conf. on Performance Evaluation Methodologies and Tools*, ser. ValueTools '08, 2008, pp. 50:1–50:8.

[13] 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.

[14] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg, "Differentiation between short and long tcp flows: predictability of the response time," in *Proc. INFOCOM*, vol. 2, March 2004, pp. 762–773.

[15] K. Avrachenkov, U. Ayesta, and P. Brown, "Batch arrival processor-sharing with application to multi-level processor-sharing scheduling," *Queueing Systems*, vol. 50, no. 4, pp. 459–480, 2005.

[16] S. Yashkov, "Mathematical problems in the theory of shared-processor systems," *Journal of Soviet Mathematics*, vol. 58, no. 2, pp. 101–147, 1992.

[17] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[18] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1976, vol. II: Computer Applications.

[19] I. Taboada, J. Fajardo, F. Liberal, and B. Blanco, "Size-based and channel-aware scheduling algorithm proposal for mean delay optimization in wireless networks," in *Proc. ICC*, June 2012, pp. 6596–6600.

[20] Y. Zhang and A. AArvidsson, "Understanding the characteristics of cellular data traffic," in *Proc. of the 2012 ACM SIGCOMM Workshop on Cellular Netw.: Operations, Challenges, and Future Design*, ser. CellNet '12, NY, USA, 2012, pp. 13–18.

[21] A. Anand and G. de Veciana, "Context-aware schedulers: Heterogeneous traffic mixes and qos/qoe trade-offs," http://users.ece.utexas.edu/~gustavo/publications.html.

[22] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed Bandit Allocation Indices*, 2nd ed. Qiley, 2011.

[23] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, vol. 2, Nov. 2003, pp. 1398–1402.