R-PF: Enhancing Service Regularity for Legacy Scheduling Policy

Abhishek Jain and I-Hong Hou
Department of ECE
Texas A&M University
College Station, TX, USA
abhishekjain1988@neo.tamu.edu and ihou@tamu.edu

***Abstract*—We present a novel scheduling policy, Regularity-Aware Proportional Fair (R-PF), that enhances service regularity for the legacy proportional fair (PF) scheduling policy. R-PF addresses the challenge that clients may have different preferences between service regularity and long-term average throughput. R-PF allows clients to specify their own preferences, and then provides services tailored to clients' specifications.**

R-PF preserves the advantages of PF, including high spectrum efficiency, fairness among clients, minimal coordination between the base station and clients, and low complexity. We analytically study the performance of R-PF under an i.i.d. channel model. We prove that each client achieves its optimal tradeoff between service regularity and throughput by reporting its true preference. We also prove that the total throughput under R-PF is almost the same as that under PF. Further, we compare the performance of R-PF against other state-of-the-art scheduling policies using trace-based simulations. Simulation results demonstrate that our policy provides the optimal performance for all kinds of clients with different preferences between service regularity and throughput. Simulation results also show that R-PF is backward compatible in that it enhances service regularity for clients who prefer high regularity without sacrificing the throughput for those who prefer high throughput.

I. Introduction

Cellular networks are being used to serve a wide range of applications, including app downloads/updates, VoIP, teleconferencing, online gaming, etc. Many of these applications, such as video streaming, generate real-time traffic that requires temporally regular services, in addition to high throughput.

There have been many studies that aim to improve the performance of clients with real-time traffic. However, the legacy proportional fair (PF) scheduling policy remains the de-facto policy for current cellular network standards [1], [2]. Although proportional fair scheduling is known to be suboptimal for serving real-time traffic [3], it has several important advantages, including high spectrum efficiency, fairness among clients, minimal coordination between base stations and clients, and low complexity. Meanwhile, most current studies can only improve the performance of real-time traffic by sacrificing some of these advantages, which is a major reason that prevent these studies from being widely adopted and implemented.

In this paper, we propose a novel scheduling policy, Regularity-Aware Proportional Fair (R-PF) scheduling, for cellular networks with fading wireless channels. In R-PF, each client may specify a single parameter that represents its preference between service regularity and throughput to the base station. Clients that do not support R-PF are assigned a default value for the parameter. The base station then provides services tailored to the preferences of clients. With the design of R-PF, clients that prefer higher service regularity are guaranteed to be scheduled more regularly, but may receive lower long-term average throughput. On the other hand, clients that prefer higher throughput indeed obtain higher throughput at the cost of less regularity. Therefore, R-PF allows clients to do fine-grained tradeoff between service regularity and throughput by themselves.

We analyze the performance of R-PF through both theoretical study and simulations based upon real-world measurements. For theoretical analysis, we consider an i.i.d. channel model. We prove that R-PF preserves the many advantages of PF mentioned above. Further, we prove that each client optimizes its performance and achieves the best tradeoff between service regularity and throughput by reporting its true preference. This feature greatly reduces the burden for clients to search for the optimal parameter to report, and makes R-PF easily implementable.

Using trace-based simulations, we demonstrate that R-PF maintains its advantages under practical wireless channels that are prone to both fast fading and slow fading. We further compare the performance of R-PF against other state-of-the-art techniques. Simulation results show that R-PF achieves near-optimal performance for both real-time clients that favor service regularity, and non-real-time clients that prefer throughput. Moreover, simulation results reveal that R-PF is backward compatible, as clients that do not support R-PF still obtain almost the same throughput under R-PF as they would under PF.

The rest of the paper is organized as follows: Section II introduces our system model and provides preliminary analysis on current scheduling policies to motivate the design of R-PF. Section III describes the design of R-PF. Section IV theoretically studies the performance of R-PF under an i.i.d. channel model. Section V presents

simulation results based on real-world measurements. Finally, Section VI concludes the paper.

II. System Model and Motivation

We consider the scenario where one base station (BS) serves N mobile clients with time-varying wireless channels. Time is slotted and numbered as $t = 1, 2, \ldots$. We use $r_n(t)$ to denote the channel capacity between client n and the BS at time t. At each time t, the BS observes the channel capacity $r_n(t)$ for all users, and then schedules one client for transmission. The scheduled client obtains $r_n(t)$ bits of data, while other clients do not obtain any data.

In many practical systems, different clients may have different preferences on service regularity and throughput. For example, clients with real-time applications, such as VoIP and video stream, are usually willing to sacrifice a little throughput in exchange to less temporal variations. On the other hand, non-real-time applications such as file transfer and data synchronization suffer little from temporal variations, and hence they have higher preference for throughput. In this paper, we study scheduling policies for serving such practical systems. We first study the performance of several widely used policies, which not only highlight the difficulty for serving clients with different preferences, but also motivate the overall design of our policy.

A popular scheduling policy that has been well-studied and widely deployed is proportional fair (PF) scheduling. Under PF, the BS tracks the long-term throughput, i.e., the average amount of data received per time slot, of each client. Let $x_n(t)$ denote the long-term throughput of client n at time t. The BS then schedules the client with the largest $r_n(t)/x_n(t)$ at time t. It has been shown that, when channel capacities can be modeled as a finite-state Markov chain, PF achieves proportional fairness by maximizing $\lim_{t\to\infty} \log x_n(t)$. Therefore, PF is usually considered to strike a good balance between total throughput and fairness.

However, PF provides no guarantees on service regularity, and may starve a client for an extended period of time. To illustrate PF's performance on throughput and service regularity, we consider a simple system where $[r_n(t)|1 \le n \le N, t = 1, 2, \ldots]$ are i.i.d. uniform random variables between $[0, 1]$. Due to symmetry, $x_n(t)$ is the same for all n at steady state, and PF simply schedules the client with the largest $r_n(t)$ at time t. We have $E\{\max_{1\le n\le N} r_n(t)\} \to 1$, as $N \to \infty$. Therefore, the total throughput $\sum_n x_n(t)$ approaches 1, and PF achieves the optimal total throughput, as the number of clients increases. On the other hand, by symmetry, the probability that client n is scheduled in any given time slot is $\frac{1}{N}$. The probability that client n is not scheduled for N consecutive time slots is $(1 - \frac{1}{N})^N \to \frac{1}{e} \approx 37\%$, as $N \to \infty$. The probability that n is not scheduled for $2N$ consecutive slots remains as high as $\frac{1}{e^2} \approx 14\%$. VoIP and video conferencing typically require more than 95% of packets to be delivered within some strict delay bounds. PF may hence fail to deliver satisfactory performance to real-time flows.

Another widely used policy is round-robin (RR). Under RR, client n is scheduled periodically at time slots $n, N + n, 2N+n, \ldots$, regardless of $r_n(t)$. RR achieves the optimal service regularity as each client is scheduled exactly once every N slots. However, the total throughput under RR is only $E\{r_n(t)\} = 0.5$.

Finally, there have been many studies that aim to provide better performance for real-time flows, including the legacy DiffServ/IntServ models [4]–[6], as well as some recent advances. Svedman, Wilson, and Ottersten [7], Kolding [8], Girici et al. [9], and Fraimis and Kotsopoulos [10] all set higher target rates for real-time flows. Li, Li, and Eryilmaz [11] enhances service regularity by giving higher priority to real-time clients that have not been served for some time. Haci, Zhu, Wang [12] combines PF with earliest-deadline-first (EDF) scheduling and gives higher priority to clients with smaller deadlines. Piro et al. [13] proposes a two-level scheduling policy that gives higher priority to real-time flows. Margolies et al. [14] imposes a hard service regularity constraint for real-time clients.

All these studies share one important design principle: They allocate more bandwidth to real-time flows than to non-real-time ones. Solutions based on this principle are therefore bound to be unfair for non-real-time flows. Further, such solutions may make non-real-time flows lie about their service preferences so as to obtain more bandwidth. Some existing mechanisms address the issue of unfairness by charging a premium for real-time clients, which further increases the complexity of the system.

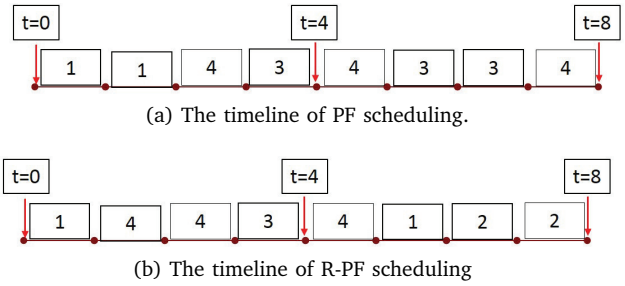
In this paper, we aim to develop a new scheduling policy for clients with different service preferences. Our policy has the following properties:

1) **Flexible:** Our policy allows clients to specify their own service preferences, and then provides services tailored to clients' specifications.
2) **Spectrum Efficient:** The total throughput under our policy is close to that under PF.
3) **Fair:** Our policy allocates the same amount of time for all clients.
4) **Tradeoff Optimal:** Each client maximizes its performance and achieves the optimal tradeoff between regularity and throughput by providing its real service preference. The performance metric of each client is formally defined in Section IV.
5) **Light weight:** Our policy has low complexity, and does not need additional infrastructure for charging clients differently.

III. Regularity-Aware Proportional Fair Scheduling

A. An Overview of the Policy

We now present our scheduling policy, Regularity-Aware Proportional Fair (R-PF) scheduling. Whenever a



(a) The timeline of PF scheduling.

(b) The timeline of R-PF scheduling

Fig. 1: An example of R-PF scheduling.

client joins the system, all clients need to announce a value called *service interval*, denoted by d_n, to the BS. As we will explain later in this section, d_n represents a client's preference between regularity and throughput. The impact of d_n will be further analyzed in Section IV.

We require that d_n to be in the form of $2^{q_n}N$ for all n, where q_n is a non-negative integer. A client can choose $d_n = \infty$ if it has no preference on regularity. In practice, the reported value d_n may not be in the form of $2^{q_n}N$. In this case, the BS increases the value of d_n so that it satisfies the constraint. Also, clients that do not support R-PF may fail to provide their service intervals. The BS chooses $d_n = \infty$ for these clients. As we will demonstrate in Section V-A, the performance of these clients under R-PF is almost identical to that under PF.

The d_n consecutive time slots in $[kd_n + 1, (k+1)d_n]$ is defined to be an *interval of* n, for any k. R-PF keeps track of the throughput that a client would have obtained under PF scheduling, and denotes this value by $x_n^*(t)$. R-PF then schedules clients using the following two principles:

1) R-PF imposes a constraint that each client n is scheduled exactly $\frac{d_n}{N}$ times in each of its interval $[kd_n + 1, (k+1)d_n]$, for all k.
2) In each time slot, R-PF schedules the client with the largest $\frac{r_n(t)}{x_n^*(t)}$ among clients that can be scheduled at time t without violating the above constraint.

Fig. 1 illustrates an example of R-PF scheduling. In this example, we assume that there are 4 clients with $d_1 = 4, d_2 = 8, d_3 = d_4 = \infty$. In Fig. 1, the top timeline is the schedule by PF, and the bottom is the one by R-PF. PF schedules client 1 at $t = 1$, and R-PF also schedules 1. At $t = 2$, PF schedules 1 again. However, R-PF cannot schedule 1 because it can only be scheduled once between $[1, 4]$. Therefore, R-PF schedules the client with the largest $\frac{r_n(2)}{x_n^*(2)}$ among clients 2,3, and 4. Suppose R-PF schedules 4 at $t = 2$. R-PF follows the schedule of PF at times $t = 3, 4, 5$, because doing so does not violate the constraint. At $t = 6$, we notice that R-PF needs to schedule client 1 once more, and client 2 twice before time $t = 8$. Therefore, R-PF only schedules clients 1 and 2 at times $t = 6, 7$, and 8, regardless of their $\frac{r_n(t)}{x_n^*(t)}$.

In the above example, we can see that a client may suffer from lower throughput due to two situations: First, even when it has the maximum $\frac{r_n(t)}{x_n^*(t)}$, it may still not be scheduled because it has already been scheduled enough

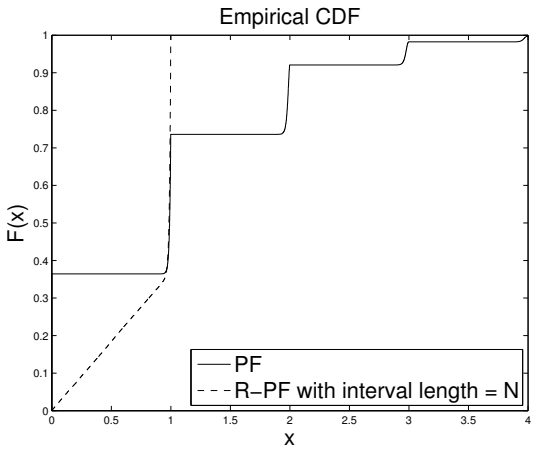


Fig. 2: CDF of the amount of data a client obtains in an interval.

times in its interval. Second, it may be scheduled even when it has a very low $\frac{r_n(t)}{x_n^*(t)}$ because we need to ensure it is scheduled enough times in an interval. Apparently, these two situations occur more frequently with smaller d_n. We can then anticipate that a client can increase its throughput by choosing larger d_n. On the other hand, it is obvious that a client with smaller d_n is scheduled more regularly. Therefore, different values of d_n reflect different preferences between throughput and regularity, and R-PF is indeed flexible in providing services tailored to clients' preferences. Further, it is also obvious that R-PF is fair because each client with $d_n < \infty$ is guaranteed to be scheduled in $\frac{1}{N}$ portion of time slots.

We use a simple simulation to illustrate the tradeoff between regularity and throughput. We consider a system with 100 clients, and $r_n(t)$ are i.i.d. random variables uniformly distributed between $[0, 1]$. We assume that all clients other than client 1 choose $d_n = \infty$. We then consider the two extreme cases where $d_1 = \infty$, under which R-PF becomes the same as PF, and $d_1 = N$. For each case, we plot the cumulative distribution function (CDF) of the amount of data client 1 obtains in each N consecutive slots. Simulation results are shown in Fig. 2. When $d_1 = \infty$, client 1 is not served at all in an interval of N slots with about 38% probability, but it also has some probability of obtaining a large amount of data in an interval. The "jumps" of the CDF correspond to the probabilities that client 1 is served 0 times, once, twice, etc. Its long-term average obtained data is about 0.99. On the other hand, when $d_1 = N$, client 1 is guaranteed to obtain some data in each interval, but the amount of data it obtained in an interval is capped by 1. As a result, its long-term average obtained data reduces to about 0.81. Therefore, by choosing $d_1 = N$, client 1 improves its service regularity at the expense of lower throughput.

B. Detailed Algorithm and Complexity Analysis

We now provide a detailed algorithm for R-PF. After obtaining all d_n, the BS sorts all clients so that $d_1 \leq d_2 \leq \ldots$.

The BS keeps track of three variables for each client n: the amount of time until the end of the client's interval, denoted by τ_n; the number of times client n should be scheduled before the end of its interval, denoted by u_n; and the total number of times that clients $1, 2, \ldots, n$ should be scheduled before the end of the interval of n, denoted by w_n.

When τ_n reaches 0, the interval of n ends and a new interval of n begins. At the beginning of an interval of n, the three variables are set to be $\tau_n = d_n$, $u_n = \frac{d_n}{N}$, and $w_n = n\frac{d_n}{N}$. In each time slot t, the BS checks if there is any client n with $\tau_n \leq w_n$. If one such client exists, the BS needs to schedule a client from $\{1, 2, \ldots, n\}$, or it becomes impossible to schedule clients in $\{1, 2, \ldots, n\}$ enough times before the end of the interval of n. In this case, the BS schedules the client with the largest $\frac{r_m(t)}{x^*_m(t)}$ from the set $\{m|1 \leq m \leq n, u_m > 0\}$. On the other hand, if $\tau_n > w_n$, for all n, the BS schedules the client with the largest $\frac{r_m(t)}{x^*_m(t)}$ from the set $\{m|u_m > 0\}$. Finally, if client n is scheduled at time t, τ_n decreases by 1, and all w_m with $m \geq n$ decreases by 1.

The complete algorithm is described in Alg. 1. It is very easy to verify that the complexity of R-PF is $\theta(N)$ per time slot. Since the complexity of PF is also $\theta(N)$, R-PF is indeed as light weight as PF.

Algorithm 1 R-PF

Sort all clients such that $d_1 \leq d_2 \leq \cdots \leq d_N$.
$\tau_n \leftarrow 0, \forall n$
for each time slot t **do**
  for $n := 1$ **to** N **do**
    if $\tau_n = 0$ **then**
      $\tau_n \leftarrow d_n$, $u_n \leftarrow \frac{d_n}{N}$, $w_n \leftarrow n\frac{d_n}{N}$
  $M \leftarrow N$
  for $n := 1$ **to** N **do**
    if $\tau_n \leq w_n$ **then**
      $M \leftarrow n$
      Break the for loop.
  $m \leftarrow \arg\max_{1 \leq n \leq M, u_n > 0} \frac{r_n(t)}{x*_n(t)}$
  Schedule client m
  $u_m \leftarrow u_m - 1$
  for $n := m$ **to** N **do**
    $w_n \leftarrow w_n - 1$
  for $n := 1$ **to** N **do**
    $\tau_n \leftarrow \tau_n - 1$
  Update $x^*_n(t)$, for all n

IV. Performance Analysis

Section III has shown that R-PF is flexible in that it provides services tailored to clients' preferences, fair in that each client is served the same portion of time, and light weight in that its complexity is as low as PF, and does not require any charging/pricing schemes. In this section, we further establish that R-PF is both spectrum efficient and tradeoff optimal.

Throughout this section, we assume that $[r_n(t)|1 \leq n \leq N, t = 1, 2, \ldots]$ are i.i.d. random variables uniformly distributed in $[0, 1]$. In Section V, we will further evaluate the performance of R-PF using traces of channel qualities from real-world experiments.

A. Spectrum Efficiency

We first study the total throughput under PF scheduling. By symmetry, every client has the same long-term average throughput. Specifically, for any two clients n and m, we have $|x^*_n(t) - x^*_m(t)| \to 0$, as $t \to \infty$, almost surely. Therefore, for any $\epsilon > 0$, as $t \to \infty$, the condition $\frac{r_n(t)}{x^*_n(t)} > \frac{r_m(t)}{x^*_m(t)} + \epsilon$ implies that $r_n(t) > r_m(t)$, almost surely[1]. As we are interested in the long-term average system performance, we can assume that PF schedules the client with the largest $r_n(t)$ in the analysis of this section. The total throughput under PF is then $E\{\max_{1 \leq n \leq N} r_n(t)\}$. We have $Prob\{\max_{1 \leq n \leq N} r_n(t) \leq z\} = Prob\{r_n(t) \leq z, \forall n\} = z^N$. Therefore, $E\{\max_{1 \leq n \leq N} r_n(t)\} = \int_0^1 (1 - z^N)dz = 1 - \frac{1}{N+1}$.

Next, we study the total throughput under R-PF scheduling. If all clients choose $d_n \equiv \infty$, then R-PF becomes the same as PF. We then focus on the other extreme case where all clients choose $d_n \equiv N$.

When $d_n \equiv N$, each client is served exactly once in each interval $[kN + 1, (k + 1)N]$, for all k. At time $kN + 1$, the BS serves the client with the largest $r_n(t)$, since $x^*_n(t)$ is the same for all clients when $t \to \infty$. The expected throughput at time $kN + 1$ is then the expected value of the largest value among N i.i.d. uniform random variables, which is $1 - \frac{1}{N+1}$. At time $kN+2$, the client that is served at time $kN + 1$ can no longer be served. Hence, the BS chooses the client with the largest $r_n(t)$ among the $(N - 1)$ remaining clients. The expected throughput at time $kN + 2$ is then the expected value of the largest value among $(N - 1)$ i.i.d. uniform random variable, which is $1 - \frac{1}{N}$. Similarly, the expected throughput at time $kN+i$ is $1 - \frac{1}{N+1-i}$. Therefore, the long-term average total throughput is $\frac{1}{N}(\sum_{i=1}^{N} 1 - \frac{1}{N+1-i}) > 1 - \frac{\ln N}{N}$.

From the analysis above, we can see that the difference of total throughputs under R-PF and under PF converges to 0 as $N \to \infty$. Fig. 3 illustrates the total throughputs of R-PF and PF.

B. Performance Metrics for Different Clients

We will show that R-PF is tradeoff optimal. We start with defining the performance metrics of different kinds of clients. To do so, we need to consider both packet generations and received services.

We divide clients into two categories: real-time clients and non-real-time clients. Non-real-time clients involve applications such as file transfer and software update. We

[1]More precisely, we have the following: For almost every sample path ω, there exists a finite value $T(\omega)$, such that the condition $\frac{r_n(t)}{x^*_n(t)} > \frac{r_m(t)}{x^*_m(t)} + \epsilon$ implies that $r_n(t) > r_m(t)$, for all $t > T(\omega)$

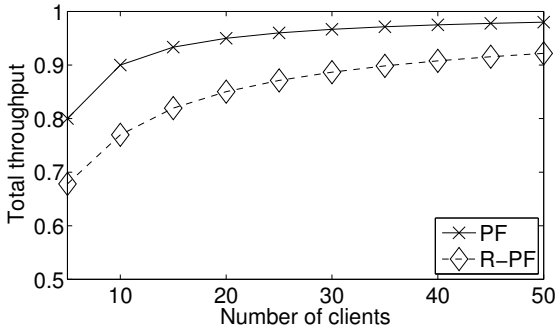


Fig. 3: Total throughput of PF and R-PF under i.i.d. channels.

assume that these clients generate saturated traffic, that is, they always have data ready for transmission. Also, the performance of non-real-time clients is measured by its long-term average throughput.

Real-time clients, on the other hand, involve applications such as online gaming, VoIP, and video streaming. These applications generate packets on-the-fly, and they have hard requirements on the delay and reliability. For example, VoIP typically requires about 95% of its packets to be delivered on time, and its delay bound is around $20ms$.

We model the behavior of real-time clients as follows: Each real-time client n generates y_n bits of data periodically with period T_n, that is, at times $1, T_n+1, 2T_n+1, \ldots$. All the y_n bits of data generated in a period need to be delivered before the end of the period, that is, within a delay bound of T_n slots. If some of the data are not delivered on time, then the client suffers from glitches in its video/audio playback. Client n requires that its video/audio playback is glitch-free in α_n of its periods.

We assume that client n employs some rate control mechanisms, such as dynamic adaptive streaming over HTTP (DASH) [15], [16], to control its value of y_n, while T_n and α_n are fixed by the application. Apparently, if y_n is too large, a lot of data cannot be delivered on time, and client n cannot enjoy α_n glitch-free periods. On the other hand, if y_n is too small, the quality of video/audio playback degrades. Therefore, client n should select the largest possible y_n subject to the constraint that the amount of data it receives in at least α_n of its periods is at least as large as y_n. We call the largest possible y_n/T_n the *α_n-reliable throughput*, which is formally defined as follows:

Definition 1: Let z_k be the amount of data that client n obtains in period $[kT_n + 1, (k + 1)T_n]$, for $k = 0, 1, 2, \ldots$. At time KT_n, the *α_n-reliable throughput* of n is $\max\{\frac{y}{T_n} | \frac{\sum_{k=0}^{K-1} 1(y \geq z_k)}{K} \geq \alpha_n\}$, where $1(\cdot)$ is the indicator function.

We use an example to illustrate the concept of α_n-reliable throughput. Suppose the amount of data that client n obtains in a period is uniformly distributed between $[0, 100]$. With probability 95%, client n obtains at least 5 units of data in a period. Therefore, as $K \to \infty$, the 95%-reliable throughput of n is 5.

C. Tradeoff Optimality

We are now ready to show that R-PF is tradeoff optimal. We assume that T_n is in the form of $2^{q_n}N$, so that it is consistent with our requirement on d_n. We also assume that all clients other than client 1 are non-real-time clients with $d_n = \infty$, and study the optimal choice of d_1 for client 1. Finally, we focus on the limit case with the number of clients goes to infinity. These assumptions will be relaxed in Section V.

Lemma 1: As $t \to \infty$, if client n has the largest $\frac{r_n(t)}{x_n^*(t)}$, then $r_n(t) \to 1$, as $N \to \infty$.

Proof: Recall that $x_n^*(t)$ is the throughput of n under PF. R-PF needs to emulate PF to obtain $x_n^*(t)$. Using a similar argument as the first paragraph of Section IV-A, we have $x_n^*(t)$ is the same for all clients when $t \to \infty$. Therefore, the client with the largest $\frac{r_n(t)}{x_n^*(t)}$ also has the largest $r_n(t)$. Further, the largest $r_n(t)$ among N i.i.d. uniform random variable converges to 1, as $N \to \infty$. ■

Theorem 1: If client 1 is a non-real-time client, it maximizes its long-term average throughput by choosing $d_1 = \infty$.

Proof: If client 1 chooses $d_1 = \infty$, then R-PF becomes the same as PF, as all clients have $d_n = \infty$. Client 1 is then only served when it has the largest $\frac{r_n(t)}{x_n^*(t)}$, and it is served $\frac{1}{N}$ of all time slots, by symmetry. Therefore, its long-term average throughput is $\frac{1}{N}$, by Lemma 1.

On the other hand, suppose client 1 chooses $d_1 < \infty$. By the design of R-PF, client 1 is only served $\frac{1}{N}$ of all time slots. Hence, its throughput cannot be larger than $\frac{1}{N}$. ■

Next, we study the case when client 1 is a real-time client. We first characterize the amount of data that client 1 can obtain under different d_1.

Let $e_n(t)$ be the indicator function that client n has the largest $\frac{r_n(t)}{x_n^*(t)}$ at time t. By Lemma 1, as $t \to \infty$, $e_n(t) = 1$ implies that $r_n(t) = 1$.

Consider the interval $[kd_1+1, (k+1)d_1]$, for some large k. Client 1 would have been scheduled $\sum_{t=kd_1+1}^{(k+1)d_1} e_1(t)$ times under PF. Recall that R-PF schedules client 1 exactly $\frac{d_1}{N}$ times during the interval $[kd_1+1, (k+1)d_1]$. Therefore, if $\sum_{t=kd_1+1}^{(k+1)d_1} e_1(t) \geq \frac{d_1}{N}$, R-PF schedules client 1 the first $\frac{d_1}{N}$ times when $e_1(t) = 1$, and client 1 obtains $\frac{d_1}{N}$ data. On the other hand, if $\sum_{t=kd_1+1}^{(k+1)d_1} e_1(t) < \frac{d_1}{N}$, R-PF schedules client 1 on each slot with $e_1(t) = 1$, as well as the last $[\frac{d_1}{N} - \sum_{t=kd_1+1}^{(k+1)d_1} e_1(t)]$ slots with $e_1(t) = 0$ in the interval. In each of the last $[\frac{d_1}{N} - \sum_{t=kd_1+1}^{(k+1)d_1} e_1(t)]$ slots, $r_1(t)$ is uniformly distributed between $[0, 1]$. Therefore, the total amount of data that client 1 obtains is $\sum_{t=kd_1+1}^{(k+1)d_1} e_1(t) + IW(\frac{d_1}{N} - \sum_{t=kd_1+1}^{(k+1)d_1} e_1(t))$, where $IW(x)$ is the sum of x i.i.d. uniform random variables, and is usually called the *Irwin-Hall* random variable. The amount of data client 1 obtains in an interval is summarized as follows:

Lemma 2: The amount of data client 1 obtains in an interval $[kd_1 + 1, (k + 1)d_1]$ is $\min\{\frac{d_1}{N}, \sum_{t=kd_1+1}^{(k+1)d_1} e_1(t)\} + IW([\frac{d_1}{N} - \sum_{t=kd_1+1}^{(k+1)d_1} e_1(t)]^+)$, as $k \to \infty$.

We now compare the α_1-reliable throughput of client 1 under the three choices $d_1 = T_1$, $d_1 < T_1$ and $d_1 = \infty$.

Theorem 2: Suppose client 1 is a real-time client with $N \leq T_1 < \infty$. For any $\alpha_1 > 50\%$, the α_1-reliable throughput under $d_1 = T_1$ is at least as large as those under $d_1 < T_1$ and $d_1 = \infty$.

Proof: We first compare the α_1-reliable throughput between all d_1 with $d_1 \leq T_1$. Since we require both d_1 and T_1 to be in the form of $2^{q_1}N$, every d_1 with $d_1 \leq T_1$ is a factor of T_1. Consider the period $[kT_1+1, (k+1)T_1]$, which consists of $\frac{T_1}{d_1}$ intervals of length d_1. Suppose $e_1(t)$ are given for all t, then the amount of obtained data is, by Lemma 2,

$$\sum_{i=0}^{T_1/d_1-1} \min\{\frac{d_1}{N}, \sum_{t=kT_1+id_1+1}^{kT_1+(i+1)d_1} e_1(t)\}$$
$$+IW(\sum_{i=0}^{T_1/d_1-1} [\frac{d_1}{N} - \sum_{t=kT_1+id_1+1}^{kT_1+(i+1)d_1} e_1(t)]^+),$$

which is stochastically smaller than the amount of obtained data under $d_1 = T_1$,

$$\min\{\frac{T_1}{N}, \sum_{t=kT_1+1}^{(k+1)T_1} e_1(t)\} + IW([\frac{T_1}{N} - \sum_{t=kT_1+1}^{(k+1)T_1} e_1(t)]^+).$$

Therefore, for any α_1, choosing $d_1 = T_1$ maximizes the α_1-reliable throughput among all $d_1 \leq T_1$.

Next, we compare the α_1-reliable throughput between $d_1 = T_1$ and $d_1 = \infty$. Given $e_1(t)$, the amount of obtained data under $d_1 = \infty$ is simply $\sum_{t=kT_1+1}^{(k+1)T_1} e_1(t)$. On the other hand, if client 1 chooses $d_1 = T_1$, its obtained data is no smaller than $\sum_{t=kT_1+1}^{(k+1)T_1} e_1(t)$, as long as $\sum_{t=kT_1+1}^{(k+1)T_1} e_1(t) \leq \frac{T_1}{N}$. Further, we have $e_1(t) = 1$ with probability $\frac{1}{N}$, and $e_1(t) = 0$ with probability $1 - \frac{1}{N}$. $\sum_{t=kT_1+1}^{(k+1)T_1} e_1(t)$ is then a Binomial random variable, whose median value is $\frac{T_1}{N}$. Hence, we have $Prob(\sum_{t=kT_1+1}^{(k+1)T_1} e_1(t) \leq \frac{T_1}{N}) \geq 50\%$, and the α_1-reliable throughput under $d_1 = T_1$ is greater or equal to that under $d_1 = \infty$, for all $\alpha_1 > 50\%$. ■

D. Extensions for More General Channel Models

Our analysis above has focused on the special case where $r_n(t)$ is uniform between $[0, 1]$, as this case yields simple expressions. However, our analysis can be easily extended to more general channel models. We now provide a brief overview on the analysis for general channel models. We only discuss the channel models and report some main results, as the detailed derivations are virtually the same as the analysis for case where $[r_n(t)]$ are i.i.d. uniform random variables.

We assume that $[r_n(t)]$ are i.i.d. random variables with some distribution. We let $F(z)$ be the inverse of the CDF of $r_n(t)$. More precisely, $F(z)$ is defined as the smallest number y such that $Prob(r_n(t) \leq y) = z$. We assume that $F(z)$ is continuous, and $F(z)$ is bounded for $z \in [0, 1]$. For example, the special case where $r_n(t)$ is uniform between $[0, 1]$ corresponds to the case where $F(z) = z$. We can then establish the following results:

For spectrum efficiency, the total throughput under PF can be calculated as $\int_0^1 NF(z)z^{N-1}dz$, and the total throughput under R-PF when all clients have $d_n \equiv N$ is $\int_0^1 F(z)\frac{\sum_{i=1}^N iz^{i-1}}{N}dz$. Given $F(z)$, we can compare the total throughputs between these two policies. For example, consider the case $F(z) = z^k$, for some k. The total throughputs under PF and R-PF are $1 - \frac{k}{N+k}$ and $1 - \frac{k\ln(N+k)/k}{N}$, respectively. Their difference converges to 0 as $N \to \infty$.

On the other hand, using very similar arguments, we can show that Theorems 1 and 2 hold for any $F(z)$, and hence R-PF remains tradeoff optimal for any i.i.d. channel models.

V. Trace-based Simulations

We present our simulations results based on real-world measurements. We use a set of traces from CRAWDAD [17]. This dataset is produced from an experiment with one sender and 16 wireless receivers. It contains measurements of received signal strength index (RSSI), in the units of dBm, for these receivers. We divide the measurements of each node into four disjoint parts, so that we have a total number of 64 parts of measurements without overlapping.

Throughout this section, we consider a LTE system with 64 clients, where each client corresponds to one part of the measurements. We assume that the system has a bandwidth of $B = 20MHz$, thermal noise of $-100dBm$, and the length of a time slot is $0.5ms$. In each slot, the channel capacity is calculated by $r_n(t) = B\log(SNR) = \frac{20}{3}$(RSSI in $dBm + 100)Mb/s$[2]. Each simulation runs for 51200 time slots, or 25.6 seconds.

A. Comparison between Policies

We compare the performance of R-PF against other state-of-the-art techniques. We implement the policies by Margolies et. al. [14], and by Li, Li, and Eryilmaz [11], with some minor modifications to fit our system model. We also implement PF scheduling. Both [14] and [11] keep track of Time-Since-Last-Service (TSLS), denoted by $\lambda_n(t)$, of each client n at each time t. $\lambda_n(t)$ can be iteratively defined as:

$$\lambda_n(t+1) = \begin{cases} 1, & \text{if client } n \text{ is served at time } t, \\ \lambda_n(t)+1, & \text{otherwise.} \end{cases}$$

In [14], each client specifies a maximum allowable TSLS, λ_n^{max}. The BS schedules a client with $\lambda_n(t) > \lambda_n^{max}$, if there is one such client. If all clients have $\lambda_n(t) \leq \lambda_n^{max}$, the BS schedules the client with the largest $\frac{r_n(t)}{x_n(t)}$, as PF does. In our implementation, we set $\lambda_n^{max} = \infty$ for non-real-time clients, and $\lambda_n^{max} = T_n$ for real-time clients. We call this policy PF-M.

[2]Some readers may note that we do not use the standard Shannon capacity: $r_n(t) = B\log(1 + SNR)$. We use $r_n(t) = B\log(SNR)$ to reduce computation overheads. Since SNR is typically larger than $20dB$ in the trace files, the error introduced by our formula is negligible.

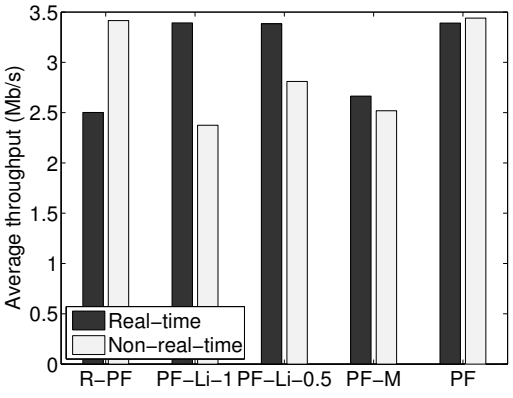
The other policy, which we call PF-Li, is based on [11]. Under PF-Li, the BS schedules the client with the largest $\frac{r_n(t)}{x_n(t)}+\beta\frac{N}{T_n}\lambda_n(t)$ in each time slot, where $T_n=\infty$ for non-real-time clients and β is a control variable. We consider the cases when $\beta=1$ and $\beta=0.5$, denoted by PF-Li-1 and PF-Li-0.5, respectively. PF-Li is also very similar to the policy proposed by Haci, Zhu, and Wang [12].

We assume that the first 32 clients are real-time clients with $T_i=64$ slots, or $32ms$, and the last 32 clients are non-real-time clients.

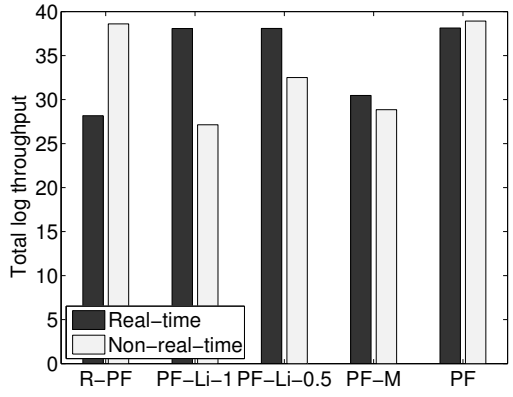
Fig. 4 plots the average throughput per client under the four policies. In order to show fairness, Fig. 4 also plots the sum of log throughput. We note that the average throughputs of non-real-time clients under PF-Li and PF-M are much smaller than that under PF. This suggests that PF-Li and PF-M hurt the performance of non-real-time clients greatly. Further, under both PF-Li and PF-M, the average throughputs of non-real-time clients are smaller than those of real-time clients. Thus, these two policies are unfair against non-real-time clients, and non-real-time clients may benefit from pretending to be real-time clients. On the other hand, the average throughput of non-real-time clients under R-PF is almost the same as that under PF, as they differ by less than 1%. This shows that R-PF is *backward compatible* as it enhances the performance of real-time clients without sacrificing non-real-time clients.

Next, we consider the performance of real-time clients. In Fig. 5, we plot the CDF of the amount of data client 1 obtains in each interval of 64 time slots. R-PF achieves the highest α_1-reliable throughput for all $\alpha_1\geq 60\%$. A closer look at the results show that, under PF, client 1 obtains near zero data in more than 70% of the intervals. This is much worse than the simple analysis in Fig. 2. This is because practical wireless channels experience both slow fading and fast fading. Therefore, the channel rates between consecutive slots can be correlated. It is then likely for a client to suffer from a poor channel rate for an extended period of time, which leads to even worse regularity under PF. Both PF-Li and PF-M significantly improve the performance of client 1 over PF, although their performance is still slightly worse than R-PF.

In summary, among the four evaluated policies, R-PF achieves near-optimal throughput for non-real-time clients, and the best α_n-reliable throughput for real-time ones.



(a) Average throughput



(b) Total log throughput

Fig. 4: Performance of real-time and non-real-time clients under different policies.

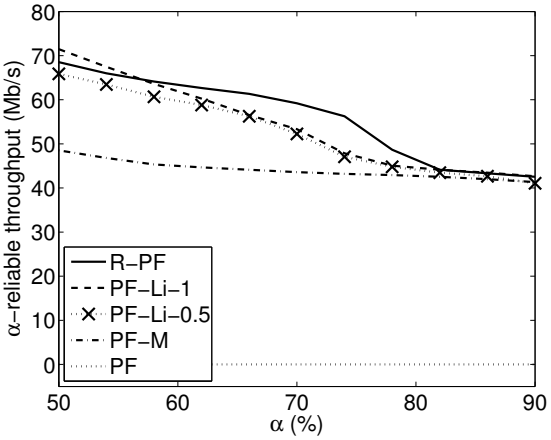
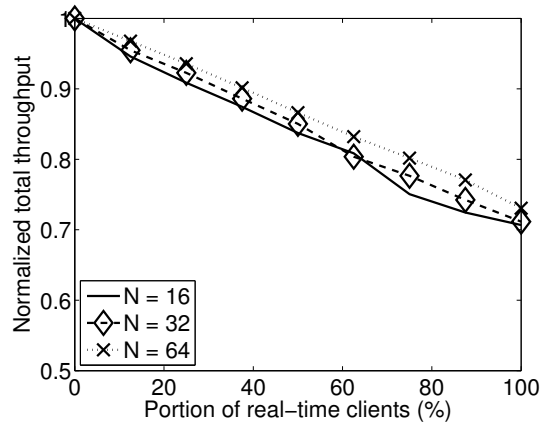


Fig. 5: α_1-reliable throughput of client 1.

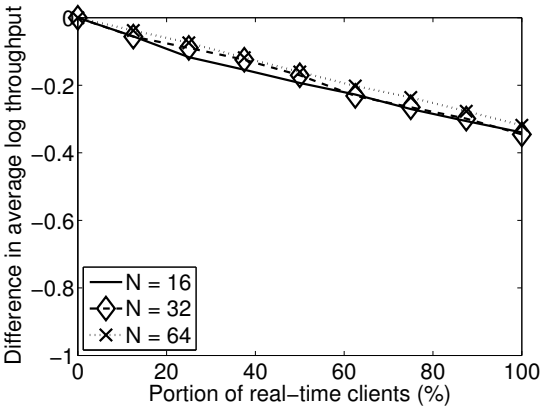
B. Spectrum Efficiency

We evaluate the spectrum efficiency of R-PF with different number of real-time clients. We consider three systems with 16 clients, 32 clients, and 64 clients, respectively. Assume that all real-time clients have $T_n=64$. For each system, we vary the portion of real-time clients and obtain the total throughput and the average log throughput of all clients. The total throughput is normalized by that under PF. The difference between the average log throughput under R-PF and PF is calculated.

Simulation results are shown in Fig. 6. Not surprisingly, the total throughput decreases with the portion of real-time clients, as R-PF becomes more constrained by regularity requirements with more real-time clients. However, the decrease in total throughput is small. With 50% real-time clients, the total throughput under R-PF remains more than 83% of that under PF. Even in the extreme case where all clients are real-time ones, the



(a) Total throughput



(b) Average log throughput

Fig. 6: Performance for different portions of real-time clients.

normalized total throughputs are still more than 70% for all three systems. The difference in average log throughput also decreases slowly with the portion of real-time clients. In particular, when all clients are real-time ones, the difference in average log throughput is smaller than -0.35. Since $e^{-0.35} \approx 70\%$, which is roughly the same as the normalized total throughput, it appears that the normalized throughput of each client is roughly the same, and R-PF remains fair for all clients.

C. Tradeoff Optimality

Finally, we verify that R-PF is tradeoff optimal by simulations. This simulation needs to run for a longer time. Therefore, we break the measurements of each wireless node into only two disjoint parts, and consider a system with 32 clients. We assume that clients 1 to 5 have $d_n = 32$, clients 6 to 10 have $d_n = 512$, and clients 11 to 15 have $d_n = \infty$. For client 16, we consider the three cases where $T_{16} = 32$, $T_{16} = 512$, and $T_{16} = \infty$. For each case, client 16 may choose d_{16} from $\{32, 128, 512, 2048, \infty\}$. If $T_{16} < \infty$, we plot the α-reliable throughput of client 16 for $\alpha = 70\%, 80\%$, and 90%, respectively. If $T_{16} = \infty$, we plot the throughput of client 16.

Simulation results are shown in Fig. 7. For all scenarios, choosing $d_{16} = T_{16}$ yields the best performance. When $T_{16} = 32$, choosing $d_{16} > 32$ results in 0 70%-reliable throughput. When $T_{16} = 512$, the α-reliable throughput gradually increases as d_{16} is increased from 32 to 512, and then abruptly drops when d_{16} becomes larger than 512. Finally, when $T_{16} = \infty$, the throughput increases with d_{16}.

VI. Conclusion

We present R-PF scheduling for cellular networks. R-PF allows clients to specify their preferences between service regularity and throughput by a single parameter. R-PF preserves many advantages of the legacy PF scheduling policy, and is backward-compatible for serving clients that do not support R-PF. This makes R-PF easily implementable. Both theoretical analysis and simulation results demonstrate that R-PF provides satisfactory performance for clients with different preferences.

There are some interesting directions for future work. The analysis in this paper has focused on i.i.d. channel models. In practice, wireless channels are prone to slow-fading, which makes channel quality in the near future depend on that in the past. In particular, a client may suffer from poor channel quality for an extended period of time, and the long-term distribution of channel quality may vary from client to client. Since R-PF is forced to serve clients regularly, its spectrum efficiency can be impacted by slow-fading. Quantifying the impact of slow-fading can be an interesting topic. On the other hand, our analysis has assumed that each client has saturated traffic, and ignored the influence of rate control protocol. Clients may employ various rate control protocols, including TCP and DASH, to adjust its traffic. In this case, it is possible that a client being scheduled by PF or R-PF does not have packets for transmission. Moreover, the traffic generated by a client depends on past system history, which, in turn, is influenced by the employed scheduling policy. Analyzing the interaction between scheduling policies and rate control protocols is an important task.

Acknowledgement

The authors would like to thank Drs. Rath Vannithamby, Jing Zhu, and Mamun Rashid from Intel for their insightful suggestions.

References

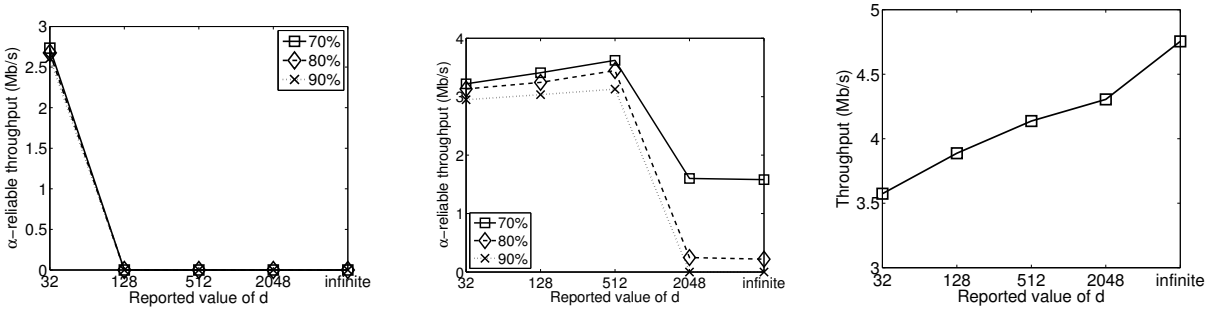
[1] H. Kim, K. Kim, Y. Han, and S. Yun, "A proportional fair scheduling for multicarrier transmission systems," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 1, pp. 409–413, IEEE, 2004.

[2] A. Toskala, H. Holma, T. Kolding, F. Frederiksen, and P. Mogensen, *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*. Wiley England, 2001.

[3] V. Vukadinovic and G. Karlsson, "Video streaming performance under proportional fair scheduling," *Selected Areas in Communications, IEEE Journal on*, vol. 28, no. 3, pp. 399–408, 2010.

[4] A. Ponnappan, L. Yang, R. Pillai, and P. Braun, "A policy based qos management system for the intserv/diffserv based internet," in *Policies for Distributed Systems and Networks, 2002. Proceedings. Third International Workshop on*, pp. 159–168, IEEE, 2002.

[5] P. Flegkas, P. Trimintzios, and G. Pavlou, "A policy-based quality of service management system for ip diffserv networks," *Network, IEEE*, vol. 16, no. 2, pp. 50–56, 2002.



(a) $T_{16} = 32$ (b) $T_{16} = 512$ (c) $T_{16} = \infty$

Fig. 7: Performance of a client for various reported d.

[6] Z. Mammeri, "Framework for parameter mapping to provide end-to-end qos guarantees in intserv/diffserv architectures," *Computer Communications*, vol. 28, no. 9, pp. 1074–1092, 2005.

[7] P. Svedman, S. K. Wilson, and B. Ottersten, "A qos-aware proportional fair scheduler for opportunistic ofdm," in *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th*, vol. 1, pp. 558–562, IEEE, 2004.

[8] T. E. Kolding, "Qos-aware proportional fair packet scheduling with required activity detection," in *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pp. 1–5, IEEE, 2006.

[9] T. Girici, C. Zhu, J. R. Agre, and A. Ephremides, "Proportional fair scheduling algorithm in ofdma-based wireless systems with qos constraints," *Communications and Networks, Journal of*, vol. 12, no. 1, pp. 30–42, 2010.

[10] I. G. Fraimis and S. A. Kotsopoulos, "Qos-based proportional fair allocation algorithm for ofdma wireless cellular systems," *Communications Letters, IEEE*, vol. 15, no. 10, pp. 1091–1093, 2011.

[11] B. Li, R. Li, and A. Eryilmaz, "Heavy-traffic-optimal scheduling with regular service guarantees in wireless networks," in *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*, pp. 79–88, ACM, 2013.

[12] H. Haci, H. Zhu, and J. Wang, "Novel scheduling for a mixture of real-time and non-real-time traffic," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 4647–4652, IEEE, 2012.

[13] G. Piro, L. A. Grieco, G. Boggia, R. Fortuna, and P. Camarda, "Two-level downlink scheduling for real-time multimedia services in lte networks," *Multimedia, IEEE Transactions on*, vol. 13, no. 5, pp. 1052–1065, 2011.

[14] R. Margolies, A. Sridharan, V. Aggarwal, R. Jana, N. Shankaranarayanan, V. A. Vaishampayan, and G. Zussman, "Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms," in *Proc. IEEE Int. Conf. on Computer Commun.(INFOCOM)*, 2014.

[15] C. Timmerer and C. Griwodz, "Dynamic adaptive streaming over http: from content creation to consumption," in *Proceedings of the 20th ACM international conference on Multimedia*, pp. 1533–1534, ACM, 2012.

[16] T. Stockhammer, "Dynamic adaptive streaming over http–: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, pp. 133–144, ACM, 2011.

[17] C. Noda, S. Prabh, M. Alves, T. Voigt, and C. A. Boano, "CRAWDAD data set cister/rssi (v. 2012-05-17)." Downloaded from http://crawdad.org/cister/rssi/, May 2012.

Abhishek Jain is a Software Engineer in OS Services Team at Apple Inc., Cupertino, USA. He received his M.S. degree in Computer Engineering from Texas A&M University in 2014 and his B.E. in Electronics & Instrumentation Engineering and M.Sc. (Hons.) in Physics from Birla Institute of Technology & Science (BITS), Pilani, India in 2011. During 2011-12, he was a Research Scholar at San Diego State University, USA. His research interests are in Wireless and Wired Networks and Operating Systems.

I-Hong Hou (S10-M12) received the B.S. in Electrical Engineering from National Taiwan University in 2004, and his M.S. and Ph.D. in Computer Science from University of Illinois, Urbana-Champaign in 2008 and 2011, respectively.

In 2012, he joined the department of Electrical and Computer Engineering at the Texas A&M University, where he is currently an assistant professor. His research interests include wireless networks, wireless sensor networks, real-time systems, distributed systems, and vehicular ad hoc networks.

Dr. Hou received the C.W. Gear Outstanding Graduate Student Award from the University of Illinois at Urbana-Champaign, and the Silver Prize in the Asian Pacific Mathematics Olympiad.