Improved TCP Performance in Wireless IP Networks through Enhanced Opportunistic Scheduling Algorithms

Thierry E. Klein, Kin K. Leung and Haitao Zheng

Abstract— **Current and next-generation wireless networks rely on multi-user diversity and scheduling techniques (such as the commonly used Proportional Fair (PF) algorithm) to achieve greater system throughput and higher efficiencies for wireless data applications over a time-varying wireless channel. In this paper, we show that the variability of the inter-scheduling intervals, as introduced by the PF scheduling algorithm, can have adverse effects on TCP and its congestion control mechanism and lead to spurious timeouts and unnecessarily low throughput. We propose an enhanced scheduling algorithm that is explicitly tuned towards throughput performance at the TCP layer. However this algorithm does not use any explicit information from the TCP layer and solely relies on information readily available at the link layer at which the scheduler resides. The performance of this improved algorithm is assessed through extensive simulations to show an average TCP throughput improvement of 12% compared to PF. In addition, the TCP-level fairness across all users is increased as is the individual user throughput.**

I. Introduction

Current wireless networks rely on multi-user diversity and scheduling techniques to achieve greater system throughputs for wireless data applications. One particularly attractive strategy is to schedule data transmissions based on the relative channel quality of the different users, while aiming for an acceptable balance between system performance and fairness among users. A well-known algorithm that achieves precisely this objective is Proportional Fair (PF) which is, for example, used in CDMA 1X EV-DO systems [1], [2]. The theoretical performance of the PF algorithm is analyzed in [3], [4], [5], [6] and references therein.

To our knowledge, most of the research work in the literature has focused on the multi-access-layer performance of the PF algorithm in particular or scheduling algorithms in general. However the impact of the scheduling algorithms on the end-to-end performance (measured at the network and transport layers) as experienced by the users' applications is not well-studied. The Transmission Control Protocol (TCP) [7] remains the most widely used transport control protocol in the Internet today, but the interaction between TCP and its congestion control mechanism and the scheduling algorithm has not yet been investigated. Our research shows that the scheduling algorithm can induce significant inter-scheduling intervals and highly varying packet transmission times. In fact the variability of the packet transmission time is shown to be large enough to cause spurious TCP timeouts, which unnecessarily trigger the TCP congestion control mechanism and reduce its throughput.

T. Klein is with the Wireless Research Lab, Bell Laboratories - Lucent Technologies, Murray Hill, NJ, USA; K. Leung is with Imperial College, London, UK; H. Zheng is with Microsoft Research, Beijing, China. This research was conducted when both K. Leung and H. Zheng were with Bell Laboratories - Lucent Technologies, Murray Hill, NJ, USA. Email: tek@lucent.com, kkleung@ieee.org, htzheng@microsoft.com

In this paper, we concentrate on the effect of PF on spurious timeouts, but note that an additional consequence of a larger packet delay variability is buffer overflow, which may lead to further throughput degradations. We propose a new scheduling algorithm that aims at controlling the variability of the inter-scheduling interval. Besides avoiding TCP timeouts, a second objective is to ensure that the TCP throughput as experienced by each user is proportionally fair to the average channel quality, thereby justifying the name of our proposed scheduling algorithm as *TCP Proportional Fair*. We emphasize that we do not allow for any information of the TCP layer to be transmitted to the multi-access (MAC) layer (at which the scheduling algorithm resides), but only rely on information that is readily available at the MAC layer and from which some of the relevant TCP parameters can be inferred. Of course further improvements may be achieved if additional cross-layer information were available. The new algorithm is motivated by analytical calculations of a queuing model with vacations [8], [9] that captures the essential effects of scheduling algorithms. Extensive simulations confirm the merits of our enhanced scheduling algorithm.

The remainder of the paper is organized as follows. In Section II we provide further motivation for our enhanced algorithm after investigating the impact of PF at the TCP layer. Section III presents our enhanced scheduling algorithm. Numerical results and some discussions are presented in Section IV. Conclusions follow in Section V.

II. Spurious Timeouts with Proportional Fair

In this section, we investigate the PF algorithm in greater detail by specifically concentrating on the resulting variability of the inter-scheduling intervals experienced by the users. Before we proceed, we briefly summarize the operations performed by the traditional PF algorithm [6]. The system is time-slotted and in time slot n, each user i reports the maximum value of the feasible transmission rate $R_i[n]$ (in bits/sec). The scheduler then determines the user with the largest scheduling metric calculated as:

$$M_i^{(MAC-PF)}[n] = \frac{R_i[n]}{A_i[n-1]}, \tag{1}$$

where $A_i[n]$ is the smoothed average throughput (in bits/sec) of user i up to time slot n and is calculated as:

$$A_i[n] = \omega\, \delta_i[n]\, R_i[n] + (1-\omega)\, A_i[n-1], \tag{2}$$

where ω is a tunable parameter and is typically taken as $\omega = \frac{1}{1000}$ [2] and $\delta_i[n] = 1$ if user i is scheduled in slot n and $\delta_i[n] = 0$ otherwise.

As an example to motivate our work, we consider a single base station serving 21 users randomly located in a cell of

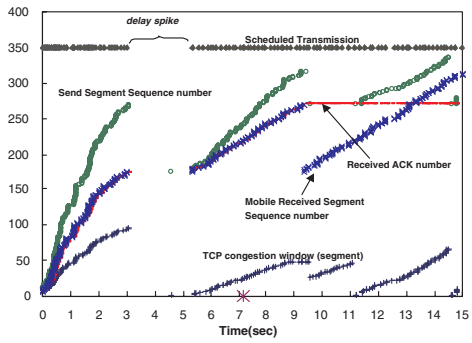

Fig. 1. Evolution of TCP congestion window under Proportional Fair.

radius 2 km and a slot duration of 5 msec. Fig. 1 shows the evolution of the TCP congestion window and the packet and acknowledgement indices over time when users are scheduled according to the PF algorithm. We also included the scheduling instants and an indication of when a frame error is recorded (for example at time 7 sec). This plot clearly exhibits that a scheduling gap on the order of 2 seconds is experienced which is significant enough to cause TCP to timeout. Since no frame errors are recorded at that time, and since we consider infinite buffers to avoid buffer overflow, the observed effects are solely due to the scheduling jitter.

TCP at the server recognizes the MAC scheduling delay spike as an instance of network congestion where all outstanding packets *i.e.* packets numbered from 176 to 270, are lost. Unaware that these packets are waiting at the base station MAC buffer, TCP times out (at time 4.6), reduces its congestion window to 1 and retransmits packets 176 to 270. Since the base station and the MAC layer do not have any capability to distinguish duplicate TCP packets, this unnecessary retransmission further degrades the throughput performance. TCP at the sender receives duplicate acknowledgments during time 9.4 and 14.8 due to duplicate packets, interprets them as a sign of packet loss and triggers the congestion control procedures to recover the packets presumed lost. This yields additional duplicate packets and leads to another TCP timeout at time 11.2. In summary, for a relatively long time after the initial delay spike, TCP continues to experience congestion and timeouts due to duplicate packet transmissions.

These results motivate the search for more efficient scheduling algorithms that control the variability of the inter-scheduling intervals. The spurious timeouts lead to TCP throughput degradation through unnecessary packet retransmissions. Hence the TCP layer throughput is no longer proportional to the MAC layer throughput and hence is not proportional to the user's channel condition, as desired by the PF algorithm. A second objective of our enhanced scheduling algorithm therefore is to achieve proportional fairness between the users at the TCP layer in a similar way that PF achieves proportional fairness at the MAC layer at which the scheduler operates.

III. Scheduling Algorithm for TCP Performance

In this section, we describe the details of our new scheduling algorithm, called *TCP Proportional Fair* (TCP-PF) algorithm. We calculate the average packet delay using a queuing-theoretic model for a system with batch arrivals and server vacations. We show how the derived expression for the average packet delay is used to determine the new scheduling metric for TCP-PF. Finally we provide the details of how TCP-PF can be implemented in recursive fashion in a practical system.

A. Theoretical Foundation of TCP-PF Algorithm

If the objective of PF is to make the MAC layer throughput proportional to the average user channel condition, our objective for the enhanced scheduler can be viewed as ensuring that the TCP layer throughput is proportionally fair. Thus, by analogy with the metric in (1), the user to be scheduled in any given time slot should be chosen as the user that maximizes the scheduling metric:

$$M_i^{(TCP-PF)}[n] = \frac{R_i[n]}{\Gamma_i[n-1]}, \tag{3}$$

where $\Gamma_i[n]$ is the smoothed average TCP throughput of user i at time slot n. In general it is very difficult to get an accurate expression for the TCP throughput as it depends on the dynamics of the congestion control mechanism, the round trip time (RTT) and its interaction with the lower layer protocols. However, for long-lived TCP flows, the throughput may be quite closely approximated as the ratio of the congestion window size to the average RTT. The congestion window is taken to be proportional to the MAC layer throughput A_i. This approximation is justified since a larger MAC throughput translates into a larger number of packets sent and acknowledgements received by the sender and therefore results in a greater increase in the TCP congestion window. The proportionality factor is assumed to be the same for all the users and therefore does not influence the scheduling metric. On the other hand, the RTT experienced by a TCP packet is the sum of several components, including the waiting, scheduling and transmission time of the packet over the air interface, processing and buffering delays in the backhaul network and at the TCP sender as well as delays in the wired network. The latter components are not easily estimated, but can be assumed to be relatively small compared to the delays incurred in the wireless part of the network. This is especially true since the air interface continues to be the bottleneck link in the network. Hence we identify the average RTT T_p experienced by a packet with the waiting and scheduling time in the transmission queue and the transmission time over the air.

In order to calculate the total packet system time, we consider an expanded $M/G/1$ queuing model [8], [9]. Specifically, the main idea is to consider that, when the scheduler allocates its resources to other users, the server takes a *vacation* for user i. *Packets* arrive according to a Poisson process of rate λ_p packets/sec. Each packet actually contains a batch of a random number of K *segments*, with mean $\overline{K}$ and second moment $\overline{K^2}$ (modelling the fragmentation of a TCP packet by the lower layers). The service time of a segment

is assumed to have a general distribution of mean $\overline{X}$ and second moment $\overline{X^2}$. However in addition, we assume that, after serving a segment, the server takes a *vacation*, with mean $\overline{V}$ and second moment $\overline{V^2}$. We assume that all service times and vacations are independent and identically distributed. The total system time $T_p(i)$ of the i-th packet is then given by:

$$T_p(i) = S(i) + \sum_{j=1}^{N(i)+K(i)-1} [X(j) + V(j)] + X(i), \quad (4)$$

where $S(i)$ is the residual service time required by the server to finish the service of the segment currently being served or an associated vacation. $N(i)$ is the number of segments waiting in queue when segment i arrives. $K(i)$ is the number of segments in the batch (*i.e.* in packet) i. $X(j)$ and $V(j)$ are, respectively the service time of segment j and the vacation taken by the server after service to segment j. Upon taking expectations in (4) and using the independence of the number of packets in the queue and the service times, the average packet system time in steady state is given by:

$$T_p = S + N_s \left(\overline{X} + \overline{V}\right) + \left(\overline{K} - 1\right) \left(\overline{X} + \overline{V}\right) + \overline{X}. \quad (5)$$

Returning to the derivation of the scheduling metric, we now have that the our enhanced scheduling metric chooses the user with the largest value of:

$$M_i^{(TCP-PF)}[n] = \frac{R_i[n]}{A_i[n-1]}\, T_i[n], \quad (6)$$

where $T_i[n]$ is the average system time of a packet of user i as estimated at time n. The remaining difficulty is to compute the packet waiting time in (5) using only information that is available at the MAC layer without explicit information transfer between the MAC and the TCP layers. The average residual time S may be calculated using similar arguments to the ones in [8]:

$$S = \frac{1}{2}\,\rho\,\frac{\overline{(X+V)^2}}{\overline{X}+\overline{V}} + \frac{1}{2}\,(1-\rho)\,\frac{\overline{V^2}}{\overline{V}}, \quad (7)$$

where the utilization factor ρ is calculated as $\rho = \lambda_p\,\overline{K}\,\left(\overline{X}+\overline{V}\right)$. If all users are assumed to be statistically identical and always have traffic to send, it can be expected that all users receive equal service time from the server (asymptotically this result has been shown for the PF scheduling algorithm [5]) and therefore we approximate a vacation as a sum of $M-1$ service times. Thus it is easily seen that $\overline{V} = (M-1)\,\overline{X}$. Similarly, a few steps of algebra show that:

$$\overline{(X+V)^2} = \frac{M}{M-1}\,\left[\overline{V^2} + \frac{1}{M-1}\,\overline{V}^2\right]. \quad (8)$$

Upon substitution into (7), we obtain that:

$$S = \frac{1}{2}\,\rho\,\frac{\overline{V^2} + \frac{1}{M-1}\,\overline{V}^2}{\overline{V}} + \frac{1}{2}\,(1-\rho)\,\frac{\overline{V^2}}{\overline{V}}, \quad (9)$$

$$\simeq \frac{1}{2}\,\frac{\overline{V_i^2}}{\overline{V_i}}, \quad (10)$$

where the last approximation is valid when the number of users M in the system is large. We have also made the dependency on user i more explicit. The term $N_s\left(\overline{X}+\overline{V}\right)$ in (5) represents the total average time required to transmit all the segments in the buffer and may be approximated by the ratio of the average queue size Q_i (in bits) at the MAC layer to the average achieved MAC throughput A_i (in bits/sec). In other words we approximate $N_s\left(\overline{X}+\overline{V}\right)$ by $\frac{Q_i}{A_i}$. Note that both quantities are available to the scheduler. The term $\left(\overline{K}-1\right)\left(\overline{X}+\overline{V}\right)$ may be ignored, especially if the average number of segments in the buffer is larger than the number of segments per TCP packet, as is typically the case in a reasonably loaded system. Finally the average transmission time of a segment $\overline{X}$ can be neglected compared to the average vacation time (again in the limit of a large number of users in the system). Thus the scheduling metric in (6) becomes:

$$M_i^{(TCP-PF)}[n] = \frac{R_i[n]}{A_i[n-1]}\,\left[\frac{1}{2}\,\frac{\overline{V_i^2}}{\overline{V_i}} + \frac{Q_i}{A_i}\right]. \quad (11)$$

We emphasize that the above scheduling metric can be viewed as an extension of the PF scheduling metric with a correction term related to the first and second moments of the inter-scheduling intervals and the expected transmission time to transmit all the data in the queue. Another physical interpretation is that the additional factor gives priority to users whose packets have experienced long waiting times, as a means to avoid TCP timeouts.

B. Implementation of TCP-PF Scheduling Algorithm

We now show how the different quantities in (11) are computed in a recursive and on-line fashion, solely based on readily available information at the MAC layer. The smoothed average throughput is calculated in (2). The average queue size (in bits) of user i is similarly calculated:

$$Q_i[n] = \omega\; q_i[n] + (1-\omega)\, Q_i[n-1], \quad (12)$$

where $q_i[n]$ is the value of the queue of user i in time slot n. The exponential weighting factor ω is the same as for the tracking of the average MAC throughput as we would like to compute both quantities using the same time scales. Let n_i^j be the time (or equivalently the slot index) in which the j-th transmission for user i was scheduled and define the inter-scheduling interval as:

$$\phi_i^j = n_i^{j+1} - n_i^j. \quad (13)$$

Whenever user i is scheduled for transmission, the smoothed values of the mean Φ_i and the second moment Σ_i of the inter-scheduling intervals are updated as follows:

$$\Phi_i^j = \alpha\,\phi_i^j + (1-\alpha)\,\Phi_i^{j-1}, \quad (14)$$

$$\Sigma_i^j = \beta\,\left(\phi_i^j\right)^2 + (1-\beta)\,\Sigma_i^{j-1}, \quad (15)$$

where α and β are tunable parameters (between 0 and 1) to control the time scales over which the respective quantities are averaged. Our numerical experience reveals that, α should be chosen relatively small to capture the long term effects of the average inter-scheduling times. Typically $\alpha = 0.1$. On the other hand, the preferred value of β is larger (typically taken as $\beta = 0.5$) to better capture the system dynamics and reflect the instantaneous changes of large inter-scheduling

intervals. The above calculations are only performed in a slot n in which user i is scheduled, and are updated after the scheduling decision has been made. In addition in every slot, the following calculations are performed before making the scheduling decision. We calculate the elapsed time since each user was last scheduled, which would correspond to the inter-scheduling interval if the user were scheduled in the current slot. Let $n_{i,last}$ be the last slot in which user i was scheduled:

$$\hat{\phi}_i[n] = n - n_{i,last}. \quad (16)$$

We now update the first and second moments of the inter-scheduling intervals for each user, as if that user were scheduled in the current slot. Specifically:

$$\hat{\Phi}_i[n] = \alpha\,\hat{\phi}_i[n] + (1-\alpha)\,\Phi_i^j, \quad (17)$$

$$\hat{\Sigma}_i[n] = \beta\left(\hat{\phi}_i[n]\right)^2 + (1-\beta)\,\Sigma_i^j, \quad (18)$$

where Φ_i^j and Σ_i^j are the current smoothed first and second moments of the inter-scheduling intervals for user i and j is the index indicating how many times user i has been scheduled. Using the expressions in (12), (17) and (18) to track Q_i, $\overline{V_i}$ and $\overline{V_i^2}$, respectively, and approximating A_i by $A_i[n-1]$, the new scheduling metric in (11) becomes:

$$M_i^{(TCP-PF)}[n] = \frac{R_i[n]}{A_i[n-1]}\left[\frac{1}{2}\frac{\hat{\Sigma}_i[n]}{\hat{\Phi}_i[n]} + \frac{Q_i[n]}{A_i[n-1]}\right]. \quad (19)$$

The scheduling metric in (19) differs from the original PF metric only by the extra factor in brackets, which is used to approximate the average RTT for the TCP performance.

IV. Numerical Results

In this section, we compare the performance of our improved algorithm to that of PF using the Opnet network simulation tool [10] to simulate a network consisting of a radio network controller, a base station and mobile terminals. Packet flows between the application server and the base station experience a fixed one-way delay of $D = 20$ msec. For simplicity we assume that the uplink channel operates at 64 kbps and 0% frame error rate. The simulated system consists of 21 users uniformly and randomly placed in a cell of radius 2 km and we simulate a typical driving speed of 30 km/h. The channel encounters frequency-flat fading and the channel estimation noise is modelled as additive white Gaussian noise with zero mean and 0.5 dB variance. Jakes model with Doppler spread corresponding to the chosen speed at a 2 GHz carrier frequency was used to simulate the Rayleigh fading. The transmission rate set consists of the following rates: 320 kbps, 480 kbps, 640 kbps, 1.28 Mbps, 1.92 Mbps and 2.56 Mbps. The rate $R_i[n]$ for user i in time slot n is selected based on SINR feedback in slot n and 1% FER requirement. Each user runs FTP over TCP/IP, requesting a file of 700 kbytes and the simulation time is 25 seconds for each chosen set of user locations. The version of TCP used is TCP Reno and the TCP packet length is fixed at 536 bytes. The granularity of the retransmission timer is chosen to be 200 msec with a minimum value of 250 msec. Finally the maximum window size is set to 64 kbytes in all of our simulations.

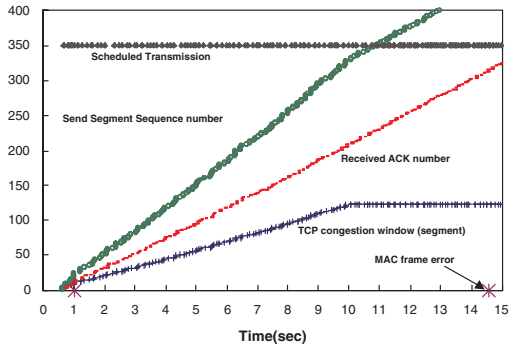

Fig. 2. Evolution of TCP congestion window under TCP-PF.

In Fig. 2, we show the evolution of the congestion window under TCP-PF for the same user in the same time interval as shown in Fig. 1 for PF. We note that TCP-PF does indeed eliminate the timeout occurrence and provides a steady transmission of TCP packets. The evolution of the congestion window is much smoother and the congestion avoidance is not unnecessarily triggered. In addition to the PF, the Round Robin (RR) and the enhanced TCP-PF algorithm, we have also considered two other algorithms. The first such algorithm, called TCP-MAX is based on the scheduling metric:

$$M_i^{(TCP-MAX)}[n] = R_i[n]\left[\frac{1}{2}\frac{\hat{\Sigma}_i[n]}{\hat{\Phi}_i[n]} + \frac{Q_i[n]}{A_i[n]}\right]. \quad (20)$$

This algorithm attempts to maximize the total system throughput as measured at the TCP layer. A second scheduler based only on the instantaneous transmission rate $R_i[n]$ attempts to maximize the MAC layer throughput, and is denoted by MAX in the remainder of the paper. In Fig. 3, we show the cumulative distribution function (cdf) for the total system throughput achieved by the different algorithms (for different simulation runs). As expected, TCP-MAX achieves a larger TCP throughput than TCP-PF, although the performance difference is not that significant. However the important result is that the throughput achieved by TCP-PF is significantly larger than that of PF (by an average of about 12%). The plot also shows that the TCP throughput achieved by MAX is severly degraded, although this algorithm achieves the largest MAC-layer throughput. The reason for this performance degradation is that the algorithm tends to only schedule users with good average channel conditions. Users in relatively poor conditions are only scheduled when they experience large channel fluctuations due to fast fading effects. However these fluctuations may not last long enough to allow a user to complete a successful packet transmission. The consequence is that the number of TCP timeouts is greatly increased, hence leading to very low TCP throughput. Finally we point out that TCP-PF largely outperforms RR, which leads us to conclude that TCP-PF does not blindly reduce the inter-scheduling intervals, but remains aware of the relative strengths of the users' channel conditions and finds a good compromise between multi-user diversity and variability of inter-scheduling intervals.

Nevertheless Fig. 3 only gives a partial analysis of the

performance of the algorithms. In Fig. 4, we show the cdf of the TCP throughput achieved by the individual users. We conclude that TCP-PF improves the throughput performance of the "high-end" users, while not degrading that of the "low-end" users. This is in stark contrast to the MAX and TCP-MAX algorithms that attempt to maximize the total system throughput at the expense of the users in relatively poor channel conditions.

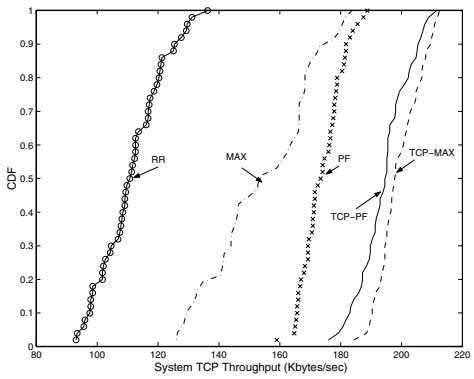

Fig. 3. CDF of TCP system throughput for different scheduling algorithms over 50 simulation runs.

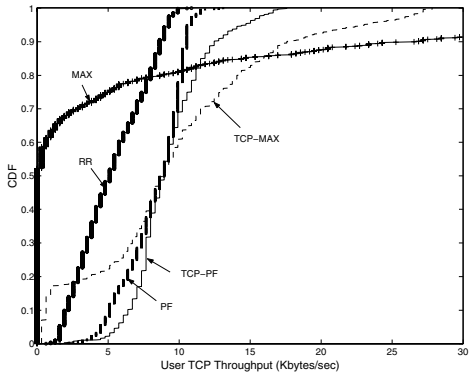

Fig. 4. CDF of individual user TCP throughput for different scheduling algorithms.

We now provide evidence that TCP-PF does in fact achieve proportional fairness at the TCP layer, as set forth as an objective for our proposed algorithm. It is well-known [6] that PF is the algorithm that maximizes the objective function $\sum_{i=1}^{M} \log\{A_i\}$, where A_i denotes the MAC-level throughput of user i. Thus we may consider the following metric to evaluate the TCP-level fairness of our algorithms:

$$F_{sys} = \frac{1}{M} \sum_{i=1}^{M} \log\{\Gamma_i\} \tag{21}$$

where Γ_i (in kbytes/sec) denotes the achieved TCP throughput of user i. The larger the metric, the closer the corresponding algorithm is deemed to achieve proportional fairness. In Fig. 5, we show the cdf of the fairness metric in (21), where the cdf is again taken over different simulation runs. We confirm that TCP-PF achieves the largest fairness metric among all algorithms.

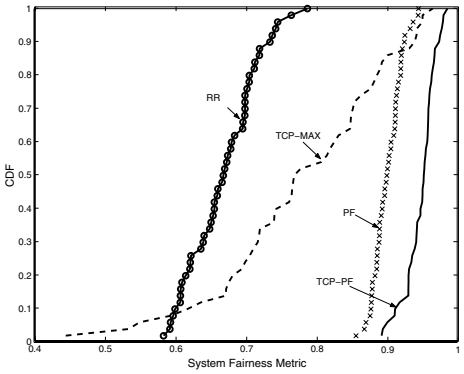

Fig. 5. CDF of the system fairness metric for different scheduling algorithms over 50 simulation runs.

In summary, TCP-PF greatly increases the system throughput compared to PF as well as the system fairness. However it is especially noteworthy that the increased system fairness in Fig. 5 is not achieved at the expense of the overall system throughput (as is evidenced by the fact that TCP-PF performs very closely to TCP-MAX in Fig. 3).

V. Conclusions

We have investigated the impact of the Proportional Fair scheduling algorithm on the performance of TCP. In particular it is demonstrated that the scheduling algorithm can lead to significant inter-scheduling intervals which may cause spurious TCP timeouts and falsely trigger the congestion control mechanism. An enhanced scheduling algorithm is proposed that alleviates these problems, avoids unnecessary TCP timeouts and consequently achieves greater TCP throughput. In addition, the enhanced algorithm provides a greater degree of fairness at the TCP layer. Finally, due to the recursive expressions, the algorithm is easily implemented without explicit information from the TCP layer, but only relies on information readily available at the multi-access layer.

References

[1] P. Bender *et al.*, "CDMA/HDR: A Bandwidth-Efficient High-Speed Wireless Data Sevice for Nomadic Users", in *IEEE Communications Magazine*, p. 70-77, July 2000.

[2] A. Jalali, R. Padovani and R. Pankaj, "Data Throughput of CDMA-HDR a High-Efficiency - High Data Rate Personal Communication Wireless System", in *Proceedings of the 50th IEEE Vehicular Technology Conference*, p. 1854-1858, 2000.

[3] S. Borst and P. Whiting, "Dynamic Rate Control Algorithms for HDR Throughput Maximization", in *Proceedings of INFOCOM 2001*, Anchorage, AK, p. 976 - 985, April 2001.

[4] S. Borst, K. Kumaran, K. Ramanan and P. Whiting, "Queuing Models for User-Level Performance of Proportional Fair Scheduling", *Technical Memorandum, Bell Laboratories - Lucent Technologies*, 2002.

[5] S. Borst, "User-Level Performance of Channel-Aware Scheduling Algorithms in Wireless Data Networks", in *Proceedings of INFOCOM 2003*, San Francisco, CA, March 2003.

[6] D. N. C. Tse, "Multiuser diversity and Proportional Fair Scheduling", *in preparation*.

[7] W. R. Stevens, *TCP/IP Illustrated, Vol. I*, Addison Wesley, 1995.

[8] D. Bertsekas and R. G. Gallager, *Data Networks*, Prentice Hall, 1992.

[9] H. Takagi, *Queueing Analysis, A Foundation of Performance Evaluation, Volume 1: Vacation and Priority Systems*, North-Holland, 1991.

[10] Opnet Technologies, http://www.opnet.com.