This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2005 proceedings.

AQuA: Aggregated Queueing Algorithm for cdma2000 Base Station Controllers

Vikas Paliwal, Biswajit Nandy, and Ioannis Lambadaris
Broadband Networks Laboratory
Carleton University, Ottawa, ON, Canada
{vpaliwal, bnandy, ioannis}@sce.carleton.ca

***Abstract*— The increasing need for enhanced data services in cellular networks requires sharing of scarce wireless channel resources among the mobile users by dynamic channel assignments. It has been shown in previous works that such a scheme causes queue management problems in the buffers shared by multiple mobiles, e.g. the input buffer at Base Station Controller (BSC), when their rates are increased after a period of lower aggregate data rate in radio links. Traditionally buffer management techniques like Random Early Detection (RED) are used for a single buffer only. In this paper, we extend the RED algorithm to an Aggregated Queueing Algorithm (AQuA) that simultaneously regulates the queueing discipline in both the shared and individual link buffers so that buffer overflow problems after aggregate link rate increase do not occur. Our algorithm relies on standard information on queueing backlog in link buffers available at BSC to perform buffer management in a unified manner with shared buffer. Furthermore, we demonstrate that our approach provides significantly greater determinism in packet transit delays over BSC and greater throughputs and similar levels of fairness as RED mechanism in shared buffers in conjunction with unregulated link buffers.**

I. Introduction

Providing high-rate data services in a cellular network poses many challenges that need to be addressed. Unlike the wired networks, where a fixed bandwidth can be dedicated to a particular user, the channel resources in wireless networks over which data can be carried, is limited in capacity and not all users can be allocated very high data rates for long durations. This limitation necessitates the need for supplemental data channels being frequently assigned and taken away from mobiles for specific durations. The information about such allocation decisions are usually available at the Base Station Controller (BSC) of the wireless networks and are usually based on the data service needs, backlog and wireless channel conditions of a particular mobile user. As an example, *cdma2000* [1] standard stipulates allocation of supplemental channels of particular rates (in range of 9.6 kb/s to 153.6 kb/s based on Rate Set 1) that can be assigned to mobiles for durations in the range of 20 ms – 5.2s. This dynamic channel assignment scheme results in fluctuations of aggregate rate of wireless links under a particular base-station controller and results in greater burstiness in the data traffic.

In a review paper [2] on ongoing research in wireless networking, the study of interconnection performance of traditional wired IP networks together with wireless links, such as those in cdma2000 networks where the bottleneck locations can change, from a queue management perspective is considered essential. In a later work [3], it was shown that variations in aggregate data rate of wireless links can have serious impacts on the performance of a cellular network. It was further shown that traditional schemes for buffer management are fairly inadequate in addressing the excessive burstiness introduced by dynamic channel assignments. Based on these works, it can be understood that a collective scheme that combines the queueing workloads in fixed rate shared buffers together with variable rate link buffers can offer improvements.

We propose that the primary target of regulating the queueing delays in a wireless network can be performed by accounting for workloads in wireless links together with other shared buffers. Such an aggregated action result in better overall system performance in terms of delays and throughputs. Further, aggregated queue management allows for suitable combination of shared and individual workloads so as to achieve desired delay targets. On the contrary, if such queue management (QM) is performed separately in these queues, parameter tuning for individual queues can not be done to yield same values of overall target metrics. An even worse situation is found in present deployments where the link buffers have no queue regulation mechanism to avoid large buildup of queues. Our solution addresses this issue by accounting for link queue term in queue management in shared queues and eliminates the need for individual QM in link buffers. We show that inclusion of link term in shared queues QM results in very stable delay performance and yields higher throughputs. In the following sections, we first introduce the problem and the proposed solution, then present an analytical model and finally present the simulation results related to this work.

II. Background

In wireless networks, as shown in figure 1, link buffers are fed with incoming data from a shared buffer. Queue management needs to be performed in both of these buffers. But current deployments rely solely on the rate controller for clearing backlog in link queues, which build huge backlog before rate controller can come into action. The link buffers have variable rates depending on the wireless conditions and data backlog. This introduces burstiness in flow of traffic going through these links. Apart from rate variability, the link buffers are usually designed with enormous sizes [2] to accommodate for retransmissions and errors. Large buffer sizes, coupled with rate variability causes severe queueing problems on other shared buffers and significant loss in throughputs as well as highly variable queueing delays can occur due to these effects.

Random Early Detection (RED) has been the most widely deployed technique in the wired network buffers. Among the RED variants adaptive version (ARED) [8] has best reported performance under varying traffic load conditions. Now onwards, we use ARED as the basic QM technique for comparison. It has been shown previously [3] that if ARED is used in conjunction with unregulated variable rate buffers, it can lead to queue overshoots. Such a situation calls for some queue management in wireless link queues as well. However direct application of RED mechanism to each of the link queues is both computationally intensive and impractical considering the small sizes of these queues. Another approach has been presented in [4] that uses deterministic dropping mechanism in each of link queues. But such mechanisms in all several hundreds of link queues will be unnecessary overheads and their efficacy will be very reduced for small link

This work was supported in part by grants from: Natural Sciences and Engineering Research Council of Canada (NSERC) and Communications and Information Technology Ontario (CITO).

 0-7803-9415-1/05/$20.00 © 2005 IEEE

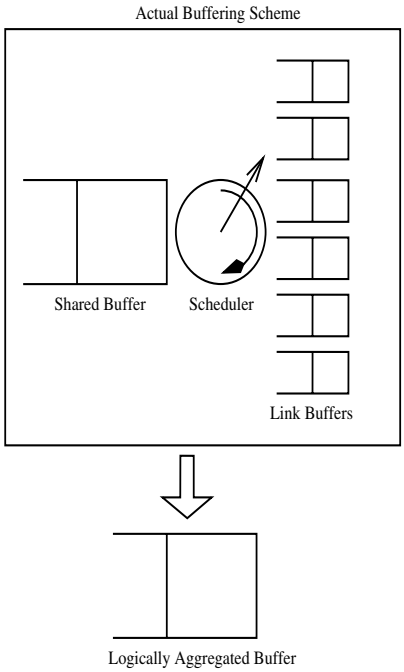


Fig. 1. Logical aggregation of shared buffer and individual link buffers at BSC into a single buffer for AQuA algorithm.

queues of merely 10-20 packets. Instead, it is our belief that aggregating the link workload with the input buffer and then performing the queue management preserves the desired statistical multiplexing and achieves significantly better performance results. Our proposal underscores the importance of performing the queue management in a unified manner. The rationale behind such an approach is that it is essentially an issue of interconnection of the two buffers and can best be managed by a unified approach that encompasses influence of queue growth in one buffer on the available buffer space in the other queue. We will see later that our mechanism helps in allowing small buffer spaces in shared buffers when link queues are large and vice versa, thereby resulting in nearly constant queueing delays under variable rate scenarios in wireless links.

III. Analysis

In this section, by means of simple relations, we show why aggregation of queues leads to better queueing delay performance. We begin with uncontrolled link buffers and then present two aggregated examples with rate variation in form of a square wave and a sinusoid. For the buffer architecture as shown in figure 1, we simplify individual link buffers into one single link buffer of variable rate $\mu^l(t)$. The shared buffer is assumed to have a fixed rate of μ^s. We assume link buffer size is fairly large to accommodate any large burst of packets. These assumptions translate to two buffers in the path of a TCP connection, with the location of bottleneck keeps on shifting depending on current value of the link rate, $\mu^l(t)$. We only consider the QM-responsive TCP flows with window size of $w(t)$ and current value of round-trip time for a packet, $r(t)$. It can be understood that the sending rate of a TCP sender at any point of time is $\frac{w(t)}{r(t)}$. For N identical sources, total instantaneous sending rate becomes $\frac{Nw(t)}{r(t)}$. Data flow arrives at the shared buffer with this rate and this governs the queue dynamics in the shared buffer. While the shared queue is empty, the arrival rate and discharge rate from shared buffer are equal. When the buffer is not empty, its rate of filling/emptying is equal to the difference of the arrival and service rates. If $q^s(t)$ is the length of the shared queue, this can be expressed as:

$$\frac{dq^s(t)}{dt} = (\frac{Nw(t)}{r(t)} - \mu^s) \cdot 1_{q^s(t)} \tag{1}$$

where $1_{q^s(t)}$ is the indicator function such that $1_{q^s(t)} = 1$ whenever $q^s(t) \geq 0$ and zero otherwise. At the link queue, the rate of arrival of packets is either $\frac{Nw(t)}{r(t)}$ or μ^s, depending on whether the shared queue is empty or not. For the empty case, a term of $((\frac{Nw(t)}{r(t)} - \mu^l(t)) \cdot 1_{q^l(t)} \cdot (1 - 1_{q^s(t)}))$, is contributed to queue dynamics where $(1 - 1_{q^s(t)})$ and $1_{q^l(t)}$ correspond to empty and non-empty states of shared and link buffers, respectively. For the case when shared queue is not empty the rate of arrival of packets at link queue is μ^s, and this contributes a term $((\mu^s - \mu^l(t)) \cdot 1_{q^l(t)} \cdot 1_{q^s(t)})$ to the link queue dynamics. Together they give rise to the relation,

$$\begin{aligned} \frac{dq^l(t)}{dt} = \; & (\frac{Nw(t)}{r(t)} - \mu^l(t)) \cdot 1_{q^l(t)} \cdot (1 - 1_{q^s(t)}) \\ & + (\mu^s - \mu^l(t)) \cdot 1_{q^l(t)} \cdot 1_{q^s(t)} \end{aligned} \tag{2}$$

Further, it can be easily seen that instantaneous value of round-trip time is a sum of round-trip propagation delay, T, and queueing delays in shared and link buffers,

$$r(t) = T + \frac{q^s(t)}{\mu^s} + \frac{q^l(t)}{\mu^l(t)}. \tag{3}$$

As mentioned earlier, we only consider QM-responsive TCP flows for our analysis. A fluid-based model for TCP was developed in [5] and can be easily verified to exactly capture the TCP dynamics in congestion-avoidance mode. It can be expressed as:

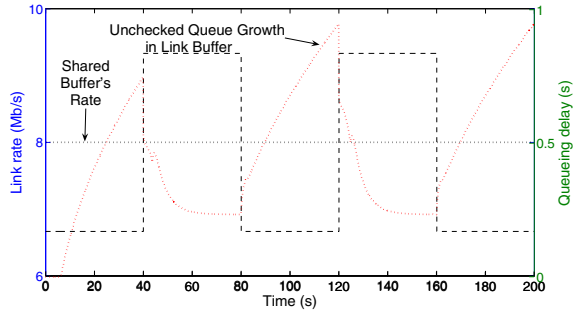
$$\frac{dw(t)}{dt} = \frac{1}{r(t)} - \frac{w^2(t)}{2r(t)} f_{QM}(t) \tag{4}$$

This model has one feed-forward term $(\frac{1}{r(t)})$ based on additive increase of window on successful receipt of ack of a transmitted packet and one feedback term $(-\frac{w^2(t)}{2r(t)} f_{QM}(t))$ that accounts for window-halving for receipt of congestion indication by means of a dropped packet or Explicit Congestion Notification (ECN) marked packet. Here $f_{QM}(t)$ can be thought of the congestion indication function as performed by the queue management technique based on one or more of the queue length and rate terms of shared and link buffers $(q^s(t), q^l(t), \mu^s, \mu^l(t))$. It can be seen that equations (1–4) completely model the cellular wireless data system under consideration.

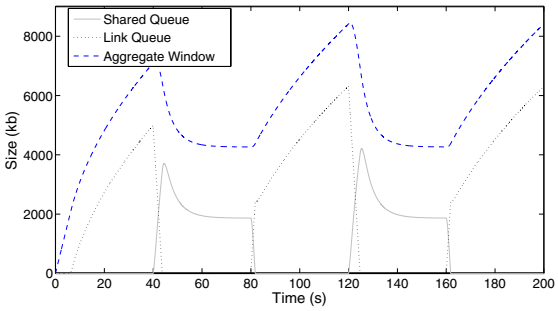
In most of the current deployments, the impact of link queues on shared buffers queues is not taken into consideration. In a variable rate scenario, this can lead to degraded performance and system underutilization [3]. The queue management function is designed to be dependent only on queue size in shared buffers. This leads to a situation under which, when the link rate is lower than shared buffers service rate and bottleneck resides at link queues, the queues in link buffers simply keep on growing. This unregulated situation ends only when supplemental channels (SCH) are assigned to mobiles so that the bottleneck again moves back to shared buffer. The rate controller in wireless data systems depend on several parameters for rate assignments and data backlog is just one of them. Besides, there can be substantial delay in the assignment of channels. All this leads to an uncontrolled system during periods of lower link rates.

Square Wave without Aggregation: As an illustrative example, if we consider a proportional controller ($f_{QM}(t) = K_p q^s(t)$) for performing the queue management function that does its drop/mark action based on only shared queue length, $q^s(t)$, it leads to an uncontrolled

system as shown in figure 2. The values of fixed parameters in this case are $N = 100, \mu^s = 8Mb/s, T = 300ms$.The link rate is varied in a square wave fashion around the shared buffer's service rate. It should be stressed that such a proportional controller has been shown to be stable for single bottleneck scenarios in previous works but becomes unstable when integrated with unregulated link queues.



(a) Delay variation with rate

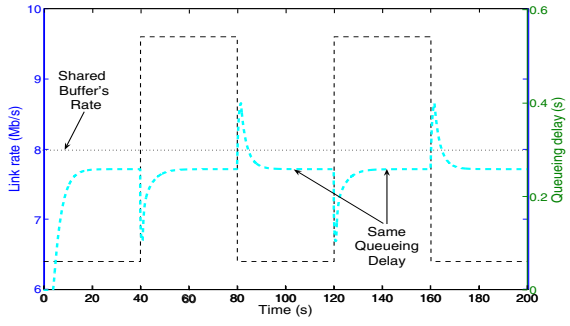


(b) Queue and window sizes

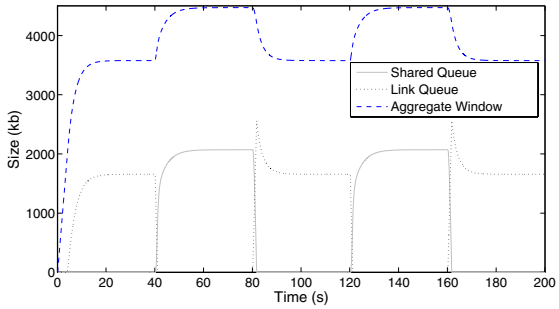
Fig. 2. Queue management with unregulated link queues.

In order to perform the queue management function in an effective manner, it is our claim that link queues should be aggregated with the shared queue. This aggregation can be done in several forms. A simplest approach would be to consider the contribution of link queues to overall queueing delays if they were located at shared buffer. This leads to a term, $\frac{\mu^s q^l(t)}{\mu^l(t)}$, which together with shared buffer's own queue becomes $q^s(t) + (\frac{\mu^s}{\mu^l(t)})q^l(t)$. Our experiments indicate that linear combination aggregations of type, $q^s(t) + (\frac{\mu^s}{\mu^l(t)})^n q^l(t)$ can help in performing QM action for a variety of link rate variations. It has to be noted that link rate variations can be obtained from empirical data and suitable aggregation can be performed at BSC. We present two examples of such aggregations.

Square Wave with Aggregation: As an example of sharp variation we consider a square wave where all the mobiles' rates simultaneously change to a new rate resulting in instantaneous jump in link rate. As an example consider the link rate variation in a square wave form as shown in figure 3, with other parameters as in figure 2. In this configuration during the under-load periods link queue becomes the bottleneck and during overload periods, the shared queue becomes the bottleneck. We assume the aggregation function to be a linear combination of form $q^s(t) + \beta q^l(t)$. The bottleneck keeps on shifting from shared buffer of rate, μ^s, and link buffer of rate, $\mu^l(t) = \mu^{ll}$, where



(a) Delay variation with rates



(b) Queue and window sizes

Fig. 3. Queue management after aggregation of queues.

μ_{ll} is the lower switched rate of square wave. For steady state operation, $(\frac{dw(t)}{dt}, \frac{dq^s(t)}{dt}, \frac{dq^l(t)}{dt} = 0)$, the system model in equations (1–4) results in:

Overload, $\mu^s < \mu^l(t)$	**Underload,** $\mu^s \geq \mu^l(t)$
$q^l(t) = 0$	$\frac{Nw_o(t)}{r_o(t)} = \mu^{ll}$
$\frac{Nw_u(t)}{r_u(t)} = \mu^s$	$q^s(t) = 0$
$r_u(t) = T + \frac{q^s}{\mu^s}$	$r_o(t) = T + \frac{q^l}{\mu^{ll}}$
$2 = w_u^2(t)K_p q^s$	$2 = w_o^2(t)K_p \beta q^l$

(5)

, where subscripts u and o refer to overload and underload conditions. Also, to meet our desired target of stable queueing delay translates to $\frac{q^s}{\mu^s} = \frac{q^l}{\mu^{ll}}$ and together with equation 5, this results in $\beta = (\frac{\mu^s}{\mu^l(t)})^3$. We use this in our aggregation and confirm this by means of simulations as shown in figure 3. Compared to the same situation in figure 2, it can be seen that aggregated queue management is able to not only regulate the queue sizes during underload periods but suitable design leads to queueing delay being tied down to a constant value.

Sinusoid with Aggregation: Next we consider a smoother variation of link rate in the form of a sine function of the type, $\mu^l(t) = (8 + 1.6\sin(2\pi\frac{t}{80}))$ Mb/s. The aggregation function that we used was of type, $q^s(t) + (\frac{\mu^s}{\mu^l(t)})q^l(t)$. The results shown in figure 4 indicate that aggregation helps in keeping the queueing delays nearly constant even when the bottlenecks keep on changing between the shared and link buffer. We proceed from these results to a generic algorithm for aggregation of queues in realistic cases where there are several hundreds of mobiles under a BSC and an equal number of link buffers for them. The following section describes the approach we present that can be

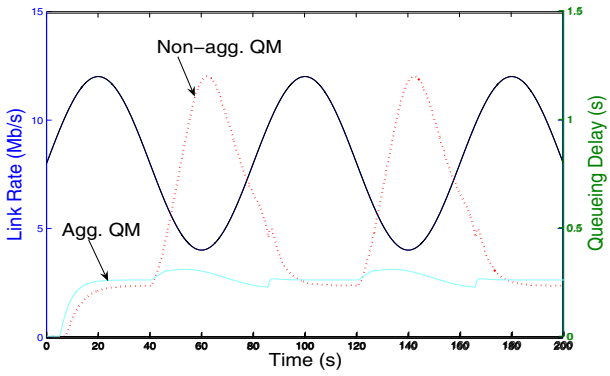


Fig. 4. Queueing delay variation for sinusoidal variation in link rate.

utilized to address the issue of burstiness due to rate variability in cellular data networks. The following section describes the approach we present that can be utilized to address the issue of burstiness due to rate variability in cellular data networks.

IV. AQuA Algorithm

This section describes the AQuA algorithm for simultaneously regulating the queues in shared input buffer and link buffers. The packet dropping criterion in this method is calculated based on not only the input buffer's queue length, rather a weighted sum of the link buffers and the shared buffers. Since the primary target of any queueing method is to avoid excessive queueing delays, we base our aggregation of the shared and input queue into a single queue as shown in figure 1 on the total delay experienced by a packet in passing through the BSC. It can be seen that this total delay is the sum of queueing delays experienced by a packet in shared buffer and its link buffer. The delay experienced by a packet in i-th link queue operating at a link rate μ_i and queue length q_i^l is, $d_i^l = (q_i^l/\mu_i^l)$. So the mean value of delay experienced by any packet in one of the N link queues under the BSC would be $\overline{d^l} = \frac{1}{N}\sum_{i=1}^{i=N}(q_i^l/\mu_i^l)$. The queue length corresponding to this mean value of delay if this buffering were in shared input buffer of rate μ^s would have been, $q^{ls} = \frac{\mu^s}{N}\sum_{i=1}^{i=N}(q_i^l/\mu_i^l)$. The total normalized queueing overload in BSC with a queue of q^s packets in shared buffer therefore would be,

$$q = q^s + \frac{\mu^s}{N}\sum_{i=1}^{i=N}(q_i^l/\mu_i^l) \tag{6}$$

It is our claim that this value of aggregated queue should be used in deciding the drop probability to perform the desired queue management. The advantage of including the link queue term with the shared queue is that during the periods of underload (when the aggregate links' rate is less than shared buffer's service rate), queues in the links do not grow up in an unregulated fashion to cause overflow problems at the shared buffer when channel allocations are performed. The measurement of link queue term can be performed at a desired rate, but since link workloads change after every slot duration for frame transmission ($slot$ =20ms for cdma2000 and 10ms for WCDMA), we propose that the link term be updated at the end of every slot duration.

The metric q that combines the queued workload in shared and link buffer helps in compensating the queueing behavior in case of abrupt changes. Essentially, if any queueing algorithm is performed on this quantity, it will eliminate any transient abrupt alternation of workloads between shared and link buffers. For a simplest drop-tail buffer, this can be set as the size of the buffer. For other mechanisms like

for each packet arrival
  calculate the average queue size avg
  if $min_{th} \leq avg_{del} < max_{th}$
    calculate probability p_a
    with probability p_a:
      mark the arriving packet
  else if $max_{th} \leq avg_{del}$
    mark the arriving packet
for every $slot$ duration
  $d_i^l \leftarrow (q_i^l/\mu_i^l)$
  $q^{ls} \leftarrow \frac{\mu^s}{N}\sum_{i=1}^{i=N} d_i^l$
where,
  $avg \leftarrow (1-w_q)avg + w_q q$
  $q \leftarrow q^s + q^{ls}$.

Fig. 5. AQuA algorithm with RED

RED this can be used as the estimate for current queueing workload in BSC. Since we observed that among the RED variants, adaptive RED (ARED) [8] has the best ability to robustly keep the queue length tied to a particular value, we suggest our AQuA modification may be applied to ARED to perform the desired queueing action. In the rest of this paper, we consider the adaptive version of RED only and often refer to it as just RED. The modified approach when applied to ARED algorithm can be written as in figure 5 (adaptive part not shown).

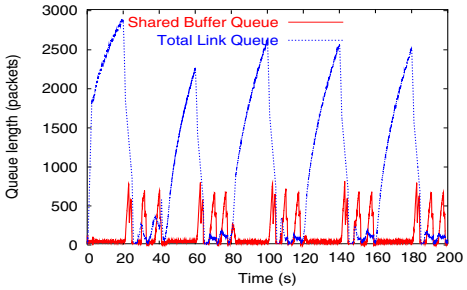
The important thing to note with this algorithm is that it allows for only a small queue in the shared buffer when the link buffers have huge workload and it allows very large queues in the shared buffer when the link queues are empty or small, which happens to be the case after increase in the aggregate rate increase. This desired behavior comes because of relating the link queue lengths with the queue management performed in the shared queues. From a practical standpoint, since the BSC has all information about a mobile's rate and data backlog, it has complete information about the individual link queue lengths, q_i^l, and mobile's rate, μ_i. The queue length measurements should be simple as BSC has an estimate of slot lengths for which it has fragmented the IP packets for a particular mobile.

V. Results

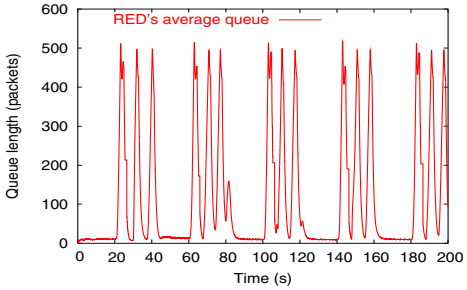
For evaluating the performance of our proposed algorithm against ARED, we performed simulation tests on four metrics of queueing delays, power curves, packet losses and fairness. For evaluating the performance of our proposed algorithm against RED, we created a similar scenario as in [3] with all the 100 mobiles changing their rates every $20s$ in a gradual fashion with a rise time of $2s$. The aggregate link rates are changed between $\pm 20\%$ of BSC's service rate. Later, AQuA's performance was found to be even better with higher swings. The thresholds for RED are set based on default setting based on shared buffer's rate (target queue = delay-bandwidth product). Other parameters for ARED (α, β) are the same as in *auto* mode of ARED implementation in [6]. For the purpose of experimental evaluation of this mechanism, modifications to RED algorithm's implementation were made in *ns2* [6] and are made available at [9]

A. Queue length behavior

The simulations are run for variety of cases and the expected behavior of AQuA performing suitable dropping/marking mechanism in

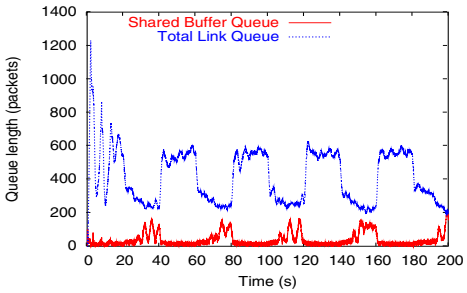


(a) Link and shared buffer queues

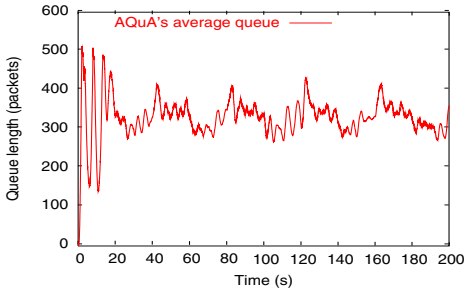


(b) Average queue

Fig. 6. Queue lengths for RED mechanism.



(a) Link and shared buffer queues



(b) Average queue

Fig. 7. Queue lengths for AQuA mechanism.

underload situations does occur and it results in keeping total workload in BSC at a fairly constant value. A sample of such simulations is shown in figures 6 and 7. Figure 6(a) shows how the entire workloads keeps on switching between the shared and the link queues. In the absence of any regulating mechanisms in the individual links, the queues simply keep on growing. Such queues immediately get transferred to the shared buffer which has a limited capacity (set as per the delay-bandwidth product) and lots of packets are lost. The RED mechanism operates in a more or less non-linear fashion with huge oscillations between the maximum limit for drop-tail behavior and zero as shown in figure 6(b). Apparently, it can be seen that the unregulated growth of queueing workload in link queues is primarily responsible for this poor performance. The only other option apart from aggregation of link and shared queues is some queue management in each of the link queues based on RED or PDPC [4] mechanisms. But such mechanisms in all several hundreds of link queues will be unnecessary overheads and their efficacy will be very reduced for small link queues of merely 10-20 packets. Instead, it is our belief that aggregating the link workload with the input buffer and then performing the queue management preserves the desired statistical multiplexing and achieves significantly better performance results.

the entire workloads keeps on switching between the shared and the link queues. In the absence of any regulating mechanisms in the individual links, the queues simply keep on growing. Such queues immediately get transferred to the shared buffer which has a limited capacity (set as per the delay-bandwidth product) and lots of packets are lost. The RED mechanism operates in a more or less non-linear fashion with huge oscillations between the maximum limit for droptail behavior and zero as shown in figure 6(b). Apparently, it can be seen that the unregulated growth of queueing workload in link queues is primarily responsible for this poor performance. Figure 7 shows the queueing behavior with same setup for AQuA mechanism. As can be seen, the workload in link queues does not grow in an indiscriminate manner. This is due to the regulatory action performed by the AQuA mechanism based random dropping/marking that accounts for the link queues. The queue in shared buffer is also kept at moderate levels together with the link queues. The most important advantage attained by AQuA algorithm is the constant value of average queue length (that is composed of both shared and link queue components) in figure 7(b). It can be seen that oscillations are significantly reduced and this results in better delay behavior for a given settings.

B. Power Curves

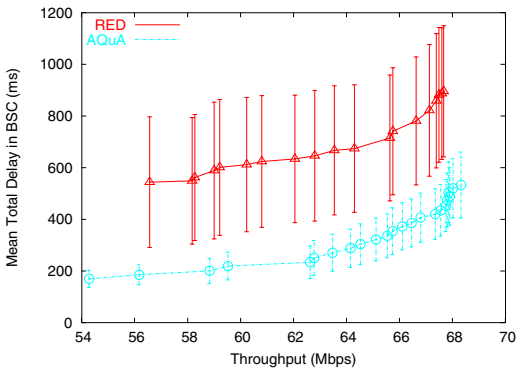


Fig. 8. Power curves for RED and AQuA

Perhaps the best way of measuring the usefulness of a queue management technique is by means of power curves as presented in [7]

that plot the throughputs against packet delays. These two are competing demands and a best match of higher throughputs with lower packet queueing delays is an important yardstick. We perform the simulations for various values of minimum threshold settings for RED and AQuA and obtain the power curves for them. The throughput in these cases are the aggregated throughputs of all the 100 mobiles under consideration and queueing delay is the total delay associated with passing of a packet through BSC's shared input and corresponding output link buffer.

Figure 8 shows these power curves. It is notable that AQuA outperforms RED in all metrics. The average queueing delay is far lesser for AQuA than that for the same value of throughput for RED. In addition to it, the variance in delay (shown by vertical bars at respective data points) is also smaller for AQuA. Based on these curves, it can be said that for conditions of oscillating bandwidths, AQuA outperforms RED in a significant manner due to its combined regulation of link queues and shared queues.

C. Packet Drops

TABLE I
PERCENTAGE OF PACKETS DROPPED

Queueing Delay (ms)	RED	AQuA
300	5.11	4.79
350	3.52	3.16
400	2.52	2.14
450	2.05	1.71
500	1.64	1.31

Packet drop profile for a queue management scheme needs to be fairly smooth, so that packets are not dropped in big bursts leading to *global synchronization*. If many packets are dropped at once, it leads to TCP timeout and under-utilization of the link. AQuA performs queue management in link queues together with the shared buffer and hence is more robust in avoiding global synchronization. On the other hand, if RED is applied only to the shared buffer, lots of packets are dropped after channel assignments. Table I shows the percentage of packets dropped for our setup. Table I shows the percentage of packets dropped for our setup. It can be seen that RED and AQuA drop nearly same number of packets. However their drop behavior differ a lot. While RED with unregulated link buffers drops most of packets after channel assignments, AQuA manages to regulate smooth and distributed drops which is important for keeping the throughput high and a nearly stable queueing delay. This is exemplified in figure 9 for one sample setup that shows AQuA's smooth drop behavior compared to RED mechanism in shared buffers only.

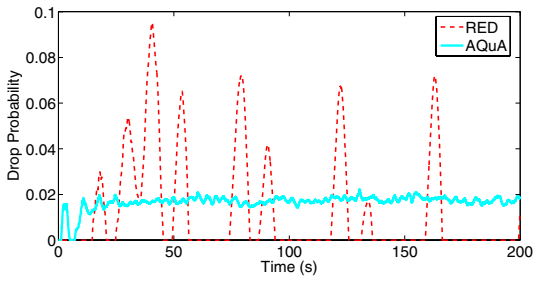


Fig. 9. Drop Profiles of RED and AQuA

D. Fairness

An ideal queue management technique, keeping other parameters the same, is ought to regulate all of the incoming flows so that each of them receive equal share of available bandwidth. In previous works it has been shown that drop-tail buffers are unfair towards certain flows that send bulk of the traffic just when the buffer gets filled. Also, some flows are lucky enough to get away with a greater share. Since the same drop-tail kind of behavior results due to sudden rate increase, shared RED buffers when combined with dumb link buffers result in an unfair system. In order to validate this hypothesis, Jain's fairness index [10], $F = \frac{(\sum_{k=1}^{k=N} \mu_i)^2}{N \sum_{k=1}^{k=N} \mu_i^2}$ was calculated for several cases. Here μ_i is the throughput received by i-th mobile. These indices are representative of degree of fairness inherent in the QM technique. These results are obtained for various values for oscillations of link rate about shared buffer's rate (8 Mb/s) and are shown in table II. It can be understood that AQuA outperforms RED due to its smooth drop behavior and offers greater fairness.

TABLE II
FAIRNESS INDICES

Rate Variation Amplitude (Mb/s)	RED	AQuA
0.5	0.98	0.99
1.0	0.91	0.97
1.5	0.83	0.92
2.0	0.76	0.89
2.5	0.68	0.86

VI. CONCLUSIONS

In this paper, we presented a method for combined management of variable rate wireless link buffers and fixed service rate shared buffers. We derived an analytical model for buffer management in cellular wireless data networks with TCP based flow control. It was shown that unregulated link buffers induce burstiness and poor performance. Further, aggregation of link queues with shared queues was shown to offer improvements for square wave and sinusoidal variations in link rates. An algorithm is presented that can be combined with ARED to address the issue and its was shown to have improved performance on metrics of queueing delays, power ratios, loss rates and fairness. We intend to verify the proposed method in a broader variety of practical deployment scenarios as a future extension to this work.

REFERENCES

[1] TIA/EIA/IS-2000.1, "Introduction to cdma2000 standards for spread spectrum systems", March 1999.
[2] A. Gurtov and S. Floyd. Modeling Wireless Links for Transport Protocols. In *ACM Comp. Comm. Review(CCR)*, pages 85–96, April 2004.
[3] V. Paliwal, P. Larijani, I. Lamabadaris, and B. Nandy, " Impact of channel variations in IP/cdma2000 interconnection performance", *to appear, ICC'2005*.
[4] M. Sagfors, R. Ludwig, M. Meyer and J. Peisa. "Queue management for TCP traffic over 3G links", *in Proc. of IEEE Wireless Communications and Networking Conference(WCNC'03)*, Mar. 2003.
[5] V. Misra, W. Gong and, D. Towsley. A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proc. of ACM SIGCOMM'00*, September 2000.
[6] Online. *http://www.isi.edu/nsnam/ns/*.
[7] S. Floyd and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", *ACM Trans. on Networking(ToN)*, vol.1 no.4, August 1993, pp. 397-413.
[8] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", Online. *http://www.icir.org/floyd/red.html*.
[9] Online. *http://www.sce.carleton.ca/vpaliwal/index_files/pub.html*.
[10] R. Jain, "The Art of Computer Systems Performance Analysis", New York: Wiley-Interscience, 1991.