Client Side Active Queue Management for 3G Cellular Networks *

Ozdemir Akin
CalIT2
University of California at San Diego
La Jolla, CA 92093
oakin@ucsd.edu

Salih Ergüt †
Ericsson, Inc.
5012 Wateridge Vista Dr.
San Diego, CA 92121
salih.ergut@ericsson.com

Ramesh R. Rao
ECE Department
University of California at San Diego
La Jolla, CA 92093
rrao@ucsd.edu

Abstract

In recent years the use of wired-cum-wireless networks has increased considerably. However, the protocols designed for wired networks does not always perform well in a wireless environment. For example, a TCP connection can easily fill-up the buffer of a slow wireless link causing other connections — such as interactive applications — experience high end-to-end delays. In this paper we develop an Active Queue Management (AQM) scheme to control the queue length at a 3G cellular base station. In contrast to regular AQM schemes our algorithm is implemented at the client node, thus we call the new algorithm Remote AQM (R-AQM). An important feature of this paper is that instead of testing our new algorithm with a software based simulator, we conducted our tests using a commercial CDMA2000 1xRTT network. Our experiments show that our algorithm can decrease the queue length considerably while maintaining very high utilization at the wireless link, hence achieves low per packet delay and jitter.

1. Introduction

In recent years increasing demand on wireless packet networks augmented the interest of both industry and academia in wired-cum-wireless networks. A typical example of a wired-cum-wireless network is where the end-user is connected to the wired packet network via a cellular or a Wi-Fi link.

Although wired and wireless links can be used together in a network, it is important to note that these links have very different characteristics. As opposed to wired links, wireless links generally have lower capacity, experience much higher bit error rates (BER), and longer delays due to fading, shadowing, power limitations, handoffs etc. In [3, 5] it has been shown that this variability in wireless link characteristics may adversely affect the performance of protocols such as TCP which are designed for wired links in mind with the assumption of low error rate and static link capacity.

Many solutions are suggested for improving the performance of TCP over wireless links. A comparison of these techniques can be found in [2]. [9] is a more recent survey about the problems and solutions of TCP over wireless links.

In this paper we focus on improving the performance of TCP flows on a 3G cellular networks — such as CDMA2000 or WCDMA — by using a new active queue management (AQM) technique. However, we implement our AQM scheme at the mobile host (MH) to regulate the queue length at the base station (BS). We call this new technique Remote-AQM (R-AQM) as we regulate the queue length at the BS remotely from a MH. In this paper we use BS in a more loose sense, and refer BS as the node between MH and the wired network.

Using active queue management schemes in a 3G environment was proposed in [7, 6, 10]. In [7] an AQM scheme is presented for a single TCP connection where packets are dropped deterministically. Via simulations it is shown that although this new algorithm outperforms RED [4] and drop-
tail in a single TCP connection scenario, its performance degrades when the number of TCP connections increases. In [6] an adaptive variation of the same scheme that modifies its parameters according to the link state is proposed.

In [10] a similar approach to [6] is used but instead of dropping packets in a deterministic manner, packets are marked using ECN in a probabilistic fashion. This algorithm also considers a single TCP connection. In all of these schemes used in [7, 6, 10] the proposed AQM algorithms are implemented at the BS.

In [8] a receiver based management algorithm is proposed for low bandwidth links. This algorithm uses the advertised window of TCP to regulate the TCP traffic. However, the algorithm assumes a constant link speed and some of the parameters requires user intervention.

The algorithm that we develop in this paper has three main features: first, as opposed to BS we implemented our method at the client node (i.e., MH). As suggested in [8], this property makes our approach easy to deploy since it does not require any modifications at the service provider equipment. Second, our scheme is not limited to single TCP connection but it can handle simultaneous multiple TCP flows. Finally, we used a commercial CDMA2000 1xRTT network to evaluate the performance of our algorithm instead of a simulator.

2. Benefits of Using an AQM Scheme at a 3G Link

In the intermediate nodes—such as routers and switches—of a network AQM schemes are used to attempt to prevent congestion and regulate the queue length by dropping packets in a proactive manner, which would eventually cause protocols such as TCP to decrease their sending rate [4, 1].

In this paper we use a new AQM scheme in a 3G environment to regulate the queue length at the BS. By regulating the queue length we mean that we try to keep the queue length steady at a small value and still have a high link utilization, which results in a small per-packet delay and delay jitter. Benefits of having a small average queue length at the BS are discussed in [7]. For completeness we will briefly mention here as it is the motivation behind using our AQM algorithm.

Long queue lengths cause the round-trip time to increase which adversely affects the performance of applications such as telnet, streaming voice or video etc. Also with high queuing delays the retransmission time-out timer of TCP increases, thus the time required from recovering from a packet loss increases.

Having a long queue may cause not only new flows joining the queue to get an unfair (low) share of the bandwidth, but also it may cause the SYN packets to time-out. During web browsing, users may switch to another webpage before the first one is fully downloaded. In this situation if the queue length is high, there would be many unnecessary packets (packets from the first web-page) in the queue that would delay the viewing of the second web-page.

Another advantage of keeping the queue length short is during handoff. If the MH moves to a new cell the state of the connection at the BS, including the packets in the buffer, must be forwarded to the new BS which increases the time to complete handoff.

In a 3G network, each MH has its own dedicated buffer at the BS which gives the opportunity to apply an AQM scheme at the MH as it can deduce all the information required to regulate the queue length at the BS, the details is discussed in the following sections.

3. Analytical Model

In this section we start with a deterministic model for a single TCP connection similar to the one used in [7, 6]. Then we use this model to generalize for multiple TCP connections.

In our analysis we assume a network topology where the client is connected to the wired network through a 3G BS and the wireless link is the bottleneck. The capacity of the forward wireless links is denoted by $C$ (packets/seconds). The round-trip propagation delays of wireless and wire-line are respectively denoted by $d_w$ and $d_{w1}$, where the total round-trip propagation delay is given by $d_0 = d_w + d_{w1}$. We also assume that $d_{w1}$ is small compared to $d_w$, thus $d_0 \approx d_w$. For notational convenience let’s define round trip delay of a packet with no queuing in the network as $d = d_0 + 1/C$.

Link layer retransmissions at BS keep the packet loss at the wireless link very low hence we neglect the packet drops both at wired and wireless links. The only packet loss is assumed to be due to buffer overflow at the BS.

3.1. Single TCP Connection

In the algorithm proposed in [7], when the queue size at the BS reaches a limit value ($q_{max}$) a single packet is dropped to slow down the TCP connection. The resulting window size behavior is similar to the one shown in Fig. 1 where $N = 1$. In [7] it is stated that in order to maintain full utilization of the wireless link $q_{max}$ must be equal to or greater than the bandwidth-delay product ($C \times d$). This can easily be proven by equating the number of packets sent in interval $[t_0, t_{A+1}]$ to $C(t_{A+1} - t_0)$ (see Appendix for the proof). In this algorithm to prevent multiple packet losses from a single sending window, after a packet is dropped $n$ packets are accepted to the queue even if the queue length exceeds $q_{max}$.
In our approach we first implement this scheme at the MH instead of BS and then we generalize it to multiple simultaneous TCP flows.

In order to implement this algorithm at the MH we need to estimate the queue length at the BS and compare this value to $q_{\text{max}}$. In our approach instead of doing this comparison we compare the round-trip delays experienced by TCP packets and compare it to a maximum delay value which is a function of $q_{\text{max}}$. This is done by using the relationship between round-trip delay experienced by the packets, $r$, and queue length at the BS, $q$, which can be written as,

$$ r = d + \frac{q}{C} \tag{1} $$

We can write a similar relationship for the maximum allowed delay, $r_{\text{max}}$, and $q_{\text{max}}$,

$$ r_{\text{max}} = d + \frac{q_{\text{max}}}{C} \tag{2} $$

Using (1) and (2) in the deterministic packet drop criteria ($q > q_{\text{max}}$) we get $r > r_{\text{max}}$ as the new inequality. As mentioned above $q_{\text{max}} = Cd$, when we plug this into (2) and use the result in inequality $r > r_{\text{max}}$ we get the new packet dropping condition as,

$$ r > 2d \tag{3} $$

To implement (3) at the client we need to estimate both delays experienced by the TCP packets and $d$ at the mobile host. $d$ can be estimated either from the three-way handshake of TCP or by pinging the server at the beginning of the connection as the buffer of the BS would be empty. In our experiments we used a static value for $d$ as the delay is mostly dominated by the delay of the wireless link.

The round-trip delays are estimated by continuously pinging the server. In order to minimize the extra load and to get a better estimate of the queue length we use the minimum packet size for ping. Another approach can be to ping the BS instead of pinging the server and use a fixed delay for $d_{\text{el}}$.

In our algorithm we use (3) to make the packet dropping decisions, i.e., whenever the observed packet delay exceeds $2d$ a packet is dropped. The effect of this packet drop will only be seen after a round-trip time (which is equal to $r$), thus after dropping a packet we do not drop any other packet for $r$ even if (3) is satisfied. A simplified pseudo code of our algorithm is given below,

```
for each received packet
    if ($r > 2d$) and (now > p_drop_timer)
        drop packet;
        p_drop_timer = now + $r$;
    endif;
```

3.2. Multiple TCP Connections

In the algorithm described above we assumed a single TCP connection. However, it is easy to see that in the case of multiple TCP connections a single packet drop only slows down one of the connections and the rest of the TCP flows continue increasing their sending rates which would increase the queue length at the BS. In order to be able to control the queue length at a multiple TCP connection scenario we modify our algorithm as follows.

Let $N$ be the number of simultaneous TCP flows. Initially, we model the queue behavior of $N$ simultaneous TCP flows as a single TCP flow that increases its window size by $N$ instead of one at every round trip-time. Also when a packet dropping decision is made at the bottleneck node (in our case the BS) one packet from each flow is dropped. A sample of the window size behavior of this model is shown in Fig. 1.

![Figure 1. TCP's congestion window (cwnd) behavior of the model used for $N$ simultaneous TCP connections. The pattern in period $[t_0, t_{A/N+1}]$ repeats itself with a period of $t_{A/N+1} - t_0$.](image)

In Appendix it is shown that $A$ must be equal to $Cd$ in order to fully utilize the link with minimum $q_{\text{max}}$ value, where $q_{\text{max}} = Cd$.

From these results we see that $q_{\text{max}}$ value is the same as the single connection case. In order to be able to use the algorithm described for the single TCP flow in a multiple TCP flow scenario we only need to modify the number of packets dropped when $r$ exceeds $2d$.

In order to prevent synchronization of all connections (i.e., all connections do not decrease or increase their $cwnd$ at the same time), we drop packets uniformly during interval $[t_0, t_{A/N+1}]$ (i.e., waiting $(t_{A/N+1} - t_0)/N$ between
successive packet drops). For notational simplicity we will call $T = t_{A/N+1} - t_0$ and $\phi = T/N$. Using the calculations in Appendix, $\phi$ can be calculated as $(3Cd'/2N + d'/2)/N$, where $d' = d$ in the ideal case. Variations to $d'$ will be explained later. It is important to note that the flows whose packets are not dropped yet would continue increasing their windows. As a result the maximum queue length may become more than the $q_{max}$ value calculated in the model. As seen in Fig. 2 the new maximum total cwnd, $\alpha$, is given by,

$$\alpha = 2B + \sum_{i=1}^{N-1} (B + \frac{iB}{N}) = 2B + \frac{3B}{2}(N-1)$$

where $B = A/N$. Then the new maximum queue length becomes $q'_{max} = \alpha - B$. Doing the similar calculations as in Appendix we get the new dropping criteria as,

$$r > \frac{d'}{C} + d = \frac{d'}{N} + \frac{3d'}{2N}(N-1) + d$$

In our model we assume that a new packet will arrive to the BS as soon as the queue is drained. However, in a real environment there can be many variations in the timing which would cause under utilization. Thus, in our experiments we target a minimum queue length of $2$ packets. Then $d'$ becomes $d + 2/C$. We estimate the bandwidth by averaging the received packets over a period of $3$ seconds. In our method $q_{max}$ is used for making the packet drop decisions and $\phi$ is only used to prevent synchronization of all connections. In the analytical model these two values are supposed to be reached simultaneously, but in a real environment there are many uncontrollable factors that would prevent this. During testing to prevent $\phi$ from being the limiting factor in packet drop decisions we multiply $\phi$ with some constant $\beta < 1$. We tried different values for $\beta$ and saw very little difference in the results. During our tests we set $\beta$ to $0.8$. The final pseudo code for multiple TCP flows is shown below,

```cpp
for each received packet
    if ($r > \frac{d'}{N} + \frac{3d'}{2N}(N-1)$ and (now > $p_{drop\_timer}$)
        drop packet;
        $p_{drop\_timer} = now + \phi * \beta$;
    endif;
```

4. Experiments

We used the setup shown in Fig. 4 where we used a commercial CDMA2000 1xRTT network including Radio Base Station (RBS), Base Station Controller (BSC) and Packet Data Serving Node (PDSN). In order to be able to get comparable results we kept the channel conditions similar at all experiments by using a wireless channel emulator (Spirent TAS4500 fader) and a noise generator (Spirent TAS4600A) for the forward link. As our main interest in this paper is the forward link, the reverse link is assumed to be perfect hence is directly connected to the BS with small attenuation. We compared our new scheme, R-AQM, with regular drop-
ran tests using packet drop. Both packet marking and dropping had very high utilization, but experiments with packet drop took more time to complete the file transfers. This is an expected result since a dropped packet has to be retransmitted and hence consume additional resources. We do not provide any results of experiments with packet drop here because of space limitations.

We ran three sets of experiments both for drop-tail and R-AQM using different link speeds, 48, 86 and 163kbps. However, these rates include all overheads of link layer packeting and retransmissions. The effective rates that we observed at the transport layer were around 38, 75 and 150 kbps respectively.

The experimental results for R-AQM and drop-tail are summarized in Table 1 and Table 2. In these tables at each cell the first value is for experiments using 48 kbps link, second is for 86 kbps and third is for 163 kbps link. In Fig. 4 we also present a sample plot for 86 kbps case. From this it can be seen that R-AQM clearly outperforms drop-tail.

It is important to note that R-AQM has an overhead as it is continuously pinging the server to estimate the queue size at the BS. The overhead of this as a percentage of the link capacity are 3%, 1.5% and 0.7% for link capacities 48, 86 and 163 kbps respectively.

From these results we see that as expected drop-tail fills the buffer and maintains very long queues all the time, which causes long queuing delays. Also it is important to note that the BS has a buffer size of 20KB, if larger buffer sizes are used the queuing delays would be higher. On the other hand R-AQM keeps the queue size at a small value while maintaining a very high utilization.

We would like to make some final remarks on the complexity of the implementation in the case of multiple TCP flows. As opposed to the mathematical analysis we did earlier we did not mark flows sequentially to comply with the derivation. Instead we marked them whenever required without checking which flow it belongs to, i.e. when measured RTT exceeded the calculated threshold. Even though there were some fluctuations on the window sizes they maintained a fair throughput with respect to each other on the average. This is because when a flow manages to skip several ECN mark rounds its window size grows larger and hence more packets from that flow are queued at the buffer. Therefore its chances of being marked at the next rounds increase drastically and hence reduces the window size. This helps maintaining fairness among competing flows and keeping the implementation simple.

<table>
<thead>
<tr>
<th>N \ R</th>
<th>BS Q.(KB)</th>
<th>RTT (sec)</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48</td>
<td>86</td>
<td>163</td>
</tr>
<tr>
<td>2</td>
<td>13.8</td>
<td>11.1</td>
<td>7.2</td>
</tr>
<tr>
<td>3</td>
<td>15.7</td>
<td>15.0</td>
<td>18.3</td>
</tr>
<tr>
<td>4</td>
<td>16.1</td>
<td>15.3</td>
<td>14.1</td>
</tr>
<tr>
<td>5</td>
<td>16.0</td>
<td>15.9</td>
<td>15.2</td>
</tr>
</tbody>
</table>

Table 1. Test results for R-AQM. BS Q. is the queue length at the BS, RTT is the average RTT and utilization is for the wireless link. N is the number of connections and R is the rate in kbps.

<table>
<thead>
<tr>
<th>N \ R</th>
<th>BS Q.(KB)</th>
<th>RTT (sec)</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>48</td>
<td>86</td>
<td>163</td>
</tr>
<tr>
<td>2</td>
<td>13.8</td>
<td>11.1</td>
<td>7.2</td>
</tr>
<tr>
<td>3</td>
<td>15.7</td>
<td>15.0</td>
<td>18.3</td>
</tr>
<tr>
<td>4</td>
<td>16.1</td>
<td>15.3</td>
<td>14.1</td>
</tr>
<tr>
<td>5</td>
<td>16.0</td>
<td>15.9</td>
<td>15.2</td>
</tr>
</tbody>
</table>

Table 2. Test results for drop-tail. BS Q. is the queue length at the BS, RTT is the average RTT and utilization is for the wireless link. N is the number of connections and R is the rate in kbps.

<table>
<thead>
<tr>
<th>N \ R</th>
<th>BS Q.(KB)</th>
<th>RTT (sec)</th>
<th>Utilization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.8</td>
<td>11.1</td>
<td>7.2</td>
</tr>
<tr>
<td>2</td>
<td>15.7</td>
<td>15.0</td>
<td>18.3</td>
</tr>
<tr>
<td>3</td>
<td>16.1</td>
<td>15.3</td>
<td>14.1</td>
</tr>
<tr>
<td>4</td>
<td>16.0</td>
<td>15.9</td>
<td>15.2</td>
</tr>
</tbody>
</table>

Figure 4. The top plot shows the BS queue length for the experiment when the link speed is 86kbps. The bottom plot shows the mean RTT for the same experiment. Note that R-AQM already achieves a mean RTT of 0.9 sec with a standard deviation of 0.2 sec, while drop-tail has a RTT of 2.0 sec on the average with a standard deviation of 0.4 sec.
5. Conclusion

In this paper we first proposed an analytical model for a single TCP connection and then extended this model to multiple TCP flows. Using this model we developed an AQM algorithm that can be implemented at the client side instead of a network node. We also provided methods for estimating the required network parameters—such as round trip time and link capacity—at the receiver with minimal overhead. We keep the BS queue length at a sufficiently small time and link capacity—at the receiver with minimal overloading the required network parameters—such as round trip time and link capacity. We compared our algorithm with drop-tail scheme using a CDMA2000 1xRTT value to maintain high link utilization. The results show that R-AQM outperforms drop-tail scheme using a CDMA2000 1xRTT value to maintain high link utilization.

A. Appendix

In Fig. 1, let \( \Delta_0 = t_1 - t_0, \Delta_1 = t_2 - t_1, \ldots \Delta_{A/N} = t_{A/N+1} - t_{A/N} \). As the pattern in the figure is periodic with a period of \( T = t_{A/N+1} - t_0, \Delta_0 = \Delta_{A/N+1}, \Delta_1 = \Delta_{A/N+2} \) etc. At \( t_{A/N+1} \) the window is halved and therefore during interval \( \Delta_{A/N+1} \) no packets are sent by the server as there are 2 A unacknowledged packets. The server has to receive acknowledgments for A packets before it can send any new packets.

Let cwnd be equal to the bandwidth delay product at \( \Delta_k \) (where \( k < A/N \)), namely \( A + Nk = Cd < 2A \). Then \( \Delta_1 = \Delta_2 = \ldots = \Delta_k = d \) since the queue is empty, and \( \Delta_{k+1} = d + N/C, \Delta_{k+2} = d + 2N/C, \ldots, \Delta_{A/N} = d + (2A - N - C)/C \). Both \( \Delta_0 = \Delta_{A/N+1} \) and \( \Delta_{A/N} \) are special cases as the acknowledgments of the 2A packets sent during \( \Delta_{A/N} \) period are received both in \( \Delta_{A/N} \) and \( \Delta_{A/N+1} \). Then \( \Delta_0 + \Delta_{A/N} = d + (2A - C)/C \).

As we are trying to fully utilize the link, the number of packets sent in period \( [t_0, t_{A/N+1}] \) must be equal to \( C(t_{A/N+1} - t_0) \). Then from Fig. 1,

\[
(A+N)+(A+2N)+\ldots+2A = C(\Delta_0 + \Delta_1 + \ldots + \Delta_{A/N})
\]

By plugging the \( \Delta_i \) values to (4) and carry out the calculations it is clear that,

\[
A = C \times d \\
\frac{k}{d} = 0
\]

is a solution.

Using (6) we can find the maximum queue length at any \( \Delta_i \) by calculating the difference between cwnd and the bandwidth delay product, which can be written as \( q = cwnd - Cd \). Then the maximum queue length in period \( [t_0, t_{A/N+1}] \) becomes,

\[
q_{\text{max}} = 2A - Cd = A
\]

References