Improving TCP Performance over 3G Links with an ACK Rate Control Algorithm

Juan J. Alcaraz, Fernando Cerdan
Department of Information Technologies and Communications
Polytechnic University of Cartagena, Plaza del Hospital, 1, 30202
Cartagena, Spain
{juan.alcaraz, fernando.cerdan}@upct.es

Abstract— **When TCP is carried over 3G cellular links, the characteristics of wireless channel and error recovey mechanisms cause overbuffering and buffer overflow at the RLC layer, degrading end-to-end performance. We propose and evaluate an algorithm that improves TCP goodput while reducing the delay. This algorithm is implemented at the RLC layer of the network side and consists of adjusting the inder-departure time of the acknowledgements (ACK) in the uplink direction, according to the congestion state of the downlink buffer. Our proposal does not requiere changes in the end user's protocol stack nor in 3GPP specifications. Additionally, it does not break TCP end-to-end semantics. In this paper we also address the parameter setting of the protocol, highlighting the importance of the radio bearer bandwith in this aspect. We illustrate how automatic reconfiguration of ACK rate control parameters avoids the degradation caused by abrupt changes in the radio bearer rate.**

Keywords-TCP over 3G, Radio Link Control, Buffer Management

I. INTRODUCTION

In 3G radio access networks, the link layer is managed by the Radio Link Control (RLC) protocol [1]. For TCP-based services, RLC is usually configured to provide a reliable service, recovering from propagation errors, thus avoiding unnecessary TCP congestion control measures.

Flows going to a single mobile terminal share a single buffer, which is expected to multiplex a number of simultaneous connections ranging form 1 to 4 TCP flows [2]. At a reliable RLC layer, upper layer packets are stored in the downlink buffer until they are fully acknowledged by the receiver side, therefore frame losses in the downlink channel result in higher RLC buffer occupancy at the network side. Considering that the current RLC specification considers a drop-tail buffer, the buffer may overflow causing consecutive packet losses. This situation is especially harmful in the first stages of a TCP connection (slow start) and has a higher impact in TCP Reno, which can only recover from consecutive losses with a Retransmission TimeOut (RTO), causing the highest reduction of the rate.

These undesired effects are even more noticeable in the presence of RB variations of the nominal rate as we show in this paper. These variations may be caused, e.g. by the 3G scheduling mechanisms or handovers among cells.

The RLC buffer should be large enough to avoid frequent overflow. However, excessive queuing cause some additional problems like Round Trip Time (RTT) inflation, unfairness between competing flows and viscous web surfing. [3].

One of the strategies to overcome these undesired interactions is the ACK Delay Control. This mechanism was presented in [4] as an algorithm aimed to improve the performance of TCP over satellite links. The underlying idea is to delay ACKs travelling through a node where its forward connection is congested. Since the ACK arrival rate determines the sending rate of new packets in a TCP source, this is a simple mechanism to reduce packet drops due to buffer overflow. In the original proposal the delay applied to ACKs was a constant value.

In this paper we propose an adaptation of this algorithm for 3G links, where the ACK departuring rate is adjusted according to the downlink queue length. We consider a general delay computing function, in which the fixed delay and the linear relation are particular cases. This generalization gives us a wider point of view in order to find accurate configurations. We provide a feasible implementation of the algorithm, and by means of extensive simulation tests, we disclose the influence of each parameter and propose configurations that clearly improve end-to-end performance compared to the conventional RLC operation.

We also show that the Radio Bearer (RB) bandwidth is one of the most relevant factors in order to find an accurate parameter setting in the proposed algorithm. Therefore, automatic parameter reconfiguration plays a crucial role in the presence of variations of the RB nominal rate.

In the 3G protocol stack, the Radio Resource Control (RRC) protocol handles the resource management algorithms, setting up and modifying layer 1 and layer 2 protocol entities. The operating scheme that we propose is completely compatible with this architecture: the RRC entity, responsible of changing the RB rate, should reconfigure the ACK rate control parameters according to the rate value. In this paper we show how the parameter reconfiguration under abrupt RB bandwidth changes benefits TCP performance.

The rest of the paper is organized as follows. Section II gives further details on the effects of 3G links characteristics on TCP performance. Section III explains our ACK rate control

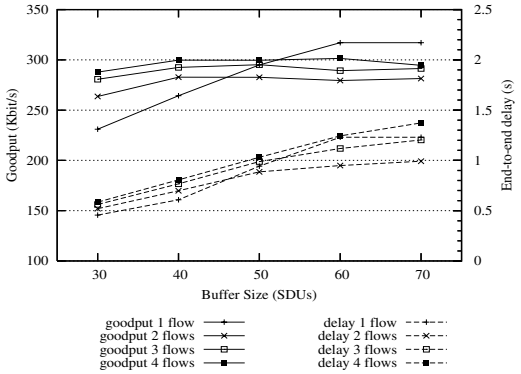

Figure 1. TCP performance over a 384 kbit/s RB.

TABLE I. SIMULATION PARAMETERS

3G link parameters	**Setting**
PDU payload size	320 bits
TTI (Transm. Time Interval)	10 ms
Transmission window	1024 PDUs
maxDAT	10
In-order-delivery	true
Status Prohibit Timer	60 ms
Missing PDU detection	true
Poll Timer	60 ms
Wireless Round Trip Delay	50 ms
Normalized doppler frequency	0,01
Poll window	50 %
Last PDU in buffer Poll	yes
Last retransmitted PDU Poll	yes
Frame Error Ratio (FER)	10%
TCP parameters	**Setting**
Maximum TCP/IP packet size	1500 bytes
Maximum allowed window	64 kbytes
Initial window	1
Wired Network Round Trip Delay	200 ms

algorithm in detail. Section IV provides a brief description of the simulation environment. Section V discusses the influence of each parameter in end-to-end performance based on extensive simulation results. The paper concludes in section VI.

II. MOTIVATION

Previous experimental [3] and simulation [5] works provide a clear view of the characteristics of 3G wireless links. The behaviour of the link buffer occupancy has shown a great impact on TCP performance. This section provides cuantitative figures and shows the effects of steep variations of the RB data rate on a TCP connection.

Fig. 1 illustrates the end-to-end goodput and delay of TCP over a radio bearer for different RLC buffer sizes. The goodput represents the successfully received packets at the receiver and the delay is the transfer time of a packet in the downlink direction, at the TCP layer. The buffer size is given in RLC Service Data Units (SDU) of 1500 bytes. Table I shows the parameter configuration for the RLC and TCP protocols. The

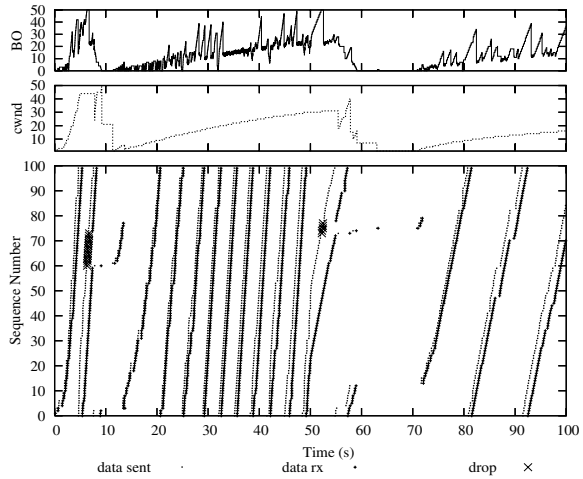

Figure 2. TCP trace over a rate-varying RB.

RLC parameters are set according to the optimizing considerations described in [6]. Further details on the simulator are provided in Section IV. As expected, Fig. 1 reveals that a larger buffer benefits the goodput performance but the overbuffering increases the latency.

The rate variation of the RB may have additional effects on TCP performance. Fig. 2 shows the trace of a TCP connection over an RB which rate starts at 384 kbit/s and switches to 128 kbit/s 50 seconds after the start time. The curve at the top shows the RLC Buffer Occupancy (*BO*). The buffer capacity is set to 50 SDUs. The TCP congestion window *cwnd* is depicted below the *BO* curve, and the curve at the bottom shows the sequence number of the TCP packets when they are sent, received and dropped.

At the first stages of the connection, multiple packets are dropped due to buffer overflow, causing an RTO in the source. The congestion window shrinks and the source begins to recover its rate slowly. The overbuffering appears again, causing high delay and higher probability of additional overflows. At $t = 50$ the buffer overflows again because of the RB rate reduction. With a lower bandwidth in the 3G link, the overbuffering increases the latency even more and causes RTO inflation.

III. PROPOSED ALGORITHM

Since the throughput of a TCP source is basically determined by the arrival rate of ACKs, it is possible to avoid overbuffering and buffer overflow in the RLC layer if this rate is conveniently adapted to the link situation. The downlink buffer *BO* is the variable used to perceive the link quality, and the rate control measures are applied only when a certain *BO* threshold (*minth*) is reached. In our algorithm, the inter-departure time, t_{id}, of ACKs traversing the RLC layer is computed with a non-linear function of *BO*, $t_{id} = f(BO)$ (1), which is obviously a continuous and only-increasing function because the departuring rate of ACKs has to be reduced when the *BO* increases.

$$t_{id} = maxd\left(\frac{BO - minth}{maxth - minth}\right)^{\alpha} \quad \text{for } BO \geq minth \text{ and } \alpha \geq 0 \qquad (1)$$

This function let us evaluate the effect of the variation rate of t_{id} respect BO. This variation rate, which is related to the "aggressiveness" of the algorithm, is determined by α (≥0). Lower values of α are tied to more aggressive delaying policies. Fig. 3 shows the shape of (1) for different values of α. In (1) it is easy to see that, if $\alpha \rightarrow 0$, *f(BO)* tends to the step function, i.e. t_{id} is constant when BO ≥ *minth*. On the other hand, if $\alpha = 1$, *f(BO)* is a linear function. Therefore, the step and the linear functions are particular cases of (1). Moreover, the absence of delay is also a particular case of this function, when $\alpha \rightarrow \infty$.

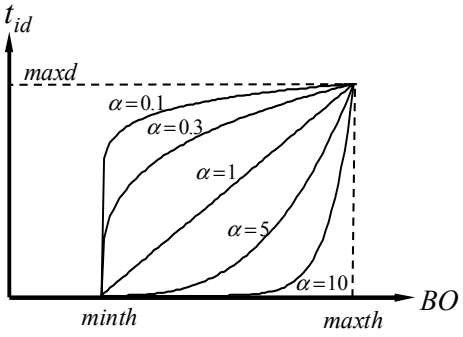

Figure 3. Representation of *f(BO)* for different α

```
for each event
   if (event == ack_arrival)
      insert ack in uplink buffer
   if (event == id_timer_expiration)
      send first ack in buffer
      LADT ← current_time
   if (uplink buffer nonempty) and (id_timer off)
      delay ← 0
      if (BO>=minth)
         t_id = f(BO)
         actual_t_id ←current_time - LADT
         if (actual_t_id < t_id)
            delay ← t_id − actual_t_id
      if (delay == 0)
         send ack
         LADT ←current_time
      else
         id_timer ← delay
         activate id_timer

Events:
ack_arrival: arrival of an upcoming ACK
id_timer_expiration: expiration of the id_timer

Saved Variables:
LADT: Last Ack Departure Time

Other:
current_time: system clock
delay: artificial delay applied to outgoing ACKs
```

Figure 4. Proposed ACK rate control algorithm.

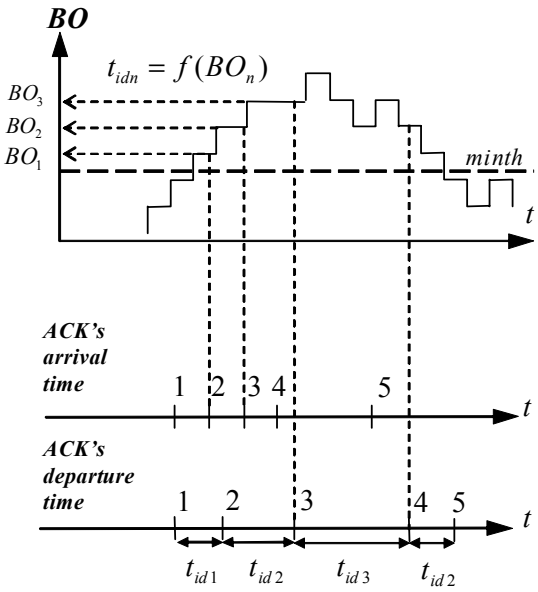

Figure 5. Illustration of the ACK rate control algorithm

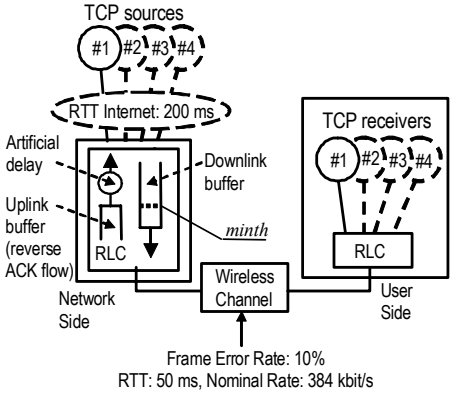

Figure 6. Simulator topology

The motivation of the variable delay strategy is to gradually adapt the source's perception of the available bandwidth and the round trip delay. Higher *BO* require higher reductions in the sending rate of the source, while for a short error burst an excessive delay on the ACKs are unnecessary.

The proposed ACK rate control algorithm is detailed in Fig. 4. The algorithm relies in the use of a single timer (*id_timer*), which delays ACKs assuring that they are spaced by t_{id}. The pseudocode identifies two different events, the arrival of an ACK packet (*ack_arrival*) and the expiration of the *id_timer* (*id_timer_expiration*). Fig. 5 illustrates the operation of the algorithm.

IV. Simulation Environment

The simulation topology, shown in Fig. 6, consists of one or several TCP sources connected to their respective receivers in the user's equipment (UE). The end-to-end connection consists of two sections: the wired network, which comprises the Internet and the 3G core network, and the radio bearer.

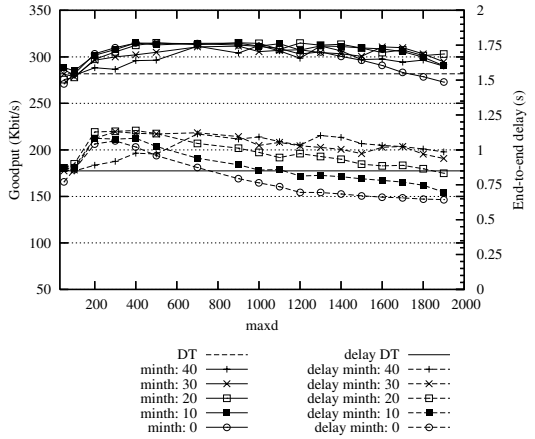

Figure 7. Performance of 1 TCP flow over a 384 Kbit/s RB

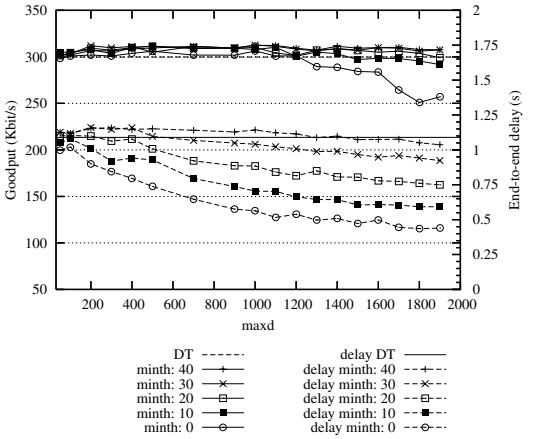

Figure 9.. Performance of 4 TCP flows over a 384 Kbit/s RB

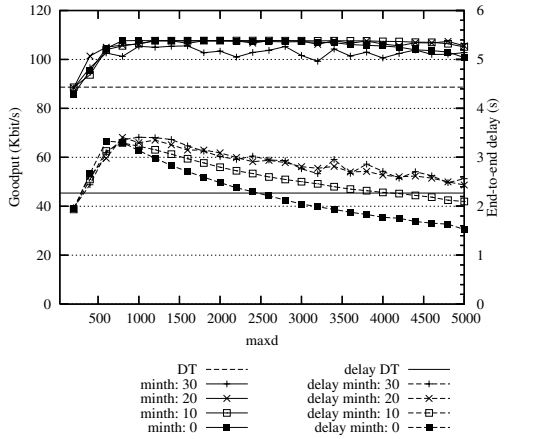

Figure 8. Performance of 1 TCP flow over a 128 Kbit/s RB

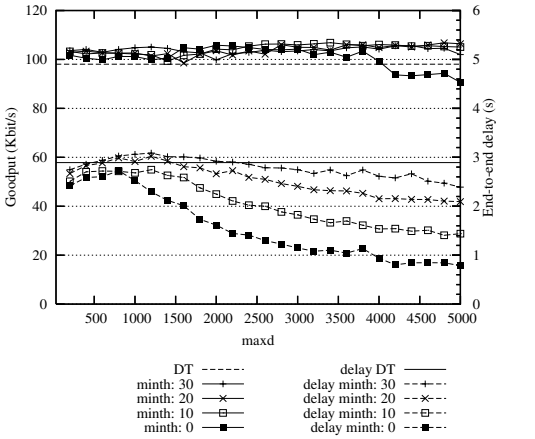

Figure 10. Performance of 4 TCP flows over a 128 Kbit/s RB

The wireless channel generates error bursts according to the model used in [7] where the Doppler frequency, f_d, of the UE determines the average burst length. The frame loss ratio is 10% in the uplink and in the downlink channels. The simulation results exposed in this paper are obtained averaging 20 runs per sample. Each run is a 60 second download session. The TCP flavour employed is TCP Reno, one of the most extended in the Internet [2].

V. PERFORMANCE EVALUATION

In order to disclose the effect of each parameter in end-to-end performance, multiple parameter combinations were tested: The values of α ranged from 0 to 40; for *minth*, the following values were evaluated: 0, 10, 20, 30 and 40 SDUs; *maxd* ranged from 50 ms to 5000 ms. The value of *maxth* equals the size of the RLC buffer, 50 SDUs.

Fig. 7 and 8 show the effect of *maxd* and *minth* in a 384 and in a 128 Kbit/s RB respectively, considering a single TCP flow and α=1. Fig. 9 and Fig. 10 correspond to the 4 TCP flows scenario. Fig. 11 and Fig. 12 show the effect of α. The goodput and delay performance for the drop-tail (DT) operation are also shown as reference values. In the single flow scenario it can be seen that, for several parameter configurations, the goodput is increased while reducing the delay respect DT figures. In the multiple-flow scenario, the goodput improvement is less noticeable but the delay reduction is up to 50%.

From Fig. 11 and Fig. 12 it seems clear that α should be lower than 1. The performance is worse for $\alpha > 1$ because the rate reduction is concentrated on higher *BO* values (see Fig. 5). The algorithm tends to react only when the congestion is too high, which is less effective preventing buffer overflow. In fact, for $\alpha >> 1$, the algorithm effect vanishes ($t_{id} = 0$).

Given a certain *BO* value, the algorithm is more aggressive if the ACK rate reduction is higher (higher t_{id}). The aggressiveness increases for lower values of *minth* and α, and for higher values of *maxd*. RBs with lower bandwidth require more aggressive configurations. The reason is that for lower nominal rates, the increment on t_{id} should be higher to obtain perceptible reductions of the ACK rate. For example, our tests show that, with a 384 Kbit/s RB, optimum configurations reduce the ACK rate, respect its nominal value, by a factor of 3 when *BO* is around 20 SDUs. Considering that one ACK is sent when two TCP packets are received in sequence, this reduction implies $t_{id} \approx$ 400 ms for $BO = 20$. In contrast, for 128 Kbit/s, a similar rate reduction is required when *BO* around 10 SDUs. This means $t_{id} \approx 1250$ ms for $BO = 10$.

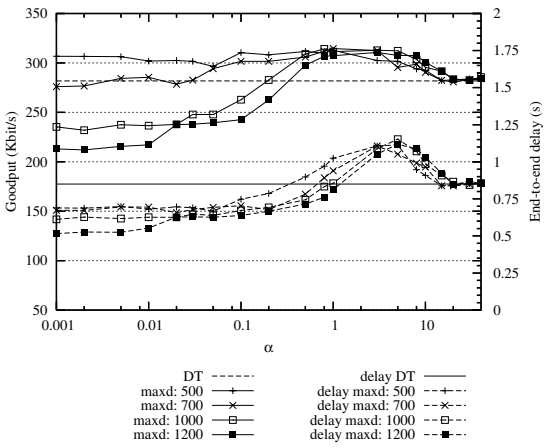

Figure 11. Effect of α in a 384 Kbit/s RB (*minth* = 10)

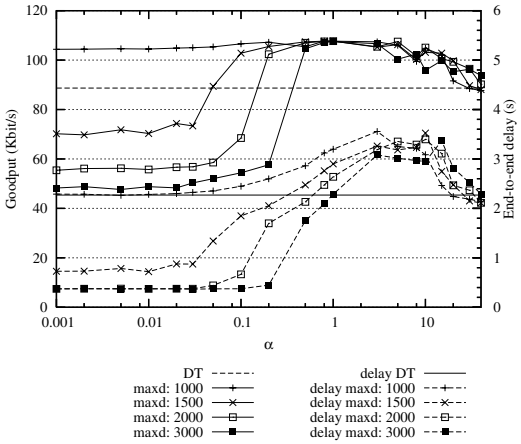

Figure 12. Effect of α in a 128 Kbit/s RB (*minth* = 5)

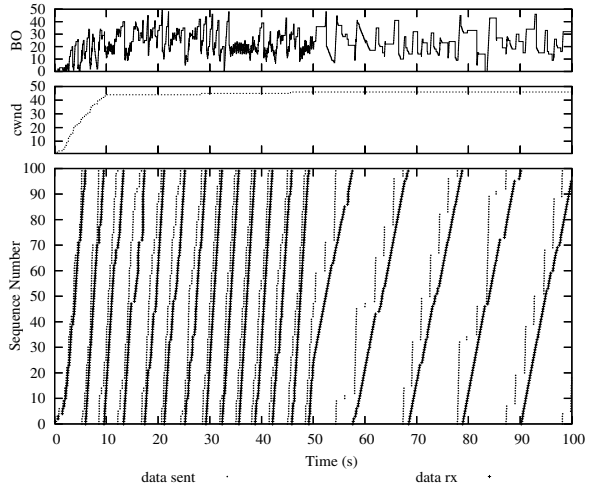

Figure 12. TCP trace over a rate-varying RB and ACK rate control automatic reconfiguration

Fig. 12 shows how an RLC buffer with automatic ACK rate control reconfiguration reacts to a sudden change in the RB rate. The example consists of a TCP flow over a 3G link that starts at 384 kbit/s and switches to 128 kbit/s after 50 seconds. For 384 Kbit/s the parameters are *minth* = 10 SDUs, *maxd* = 500 ms and $\alpha = 0.2$. When the RRC modifes the RB rate, two parameters are automatically changed: *minth* = 0 and *maxd* = 2000 ms, successfully avoiding further overbuffering.

VI. CONCLUSION

Our proposed ACK rate control algorithm can improve end-to-end goodput and delay performance when RLC supports a single or multiple TCP flows. By means of extensive simulation tests we provide insight about the influence of each parameter, which let us find several accurate configurations. The algorithm performs equally well in the slow start and in the congestion avoidance phase of TCP connections, and is useful for a single or several concurrent flows. In contrast to similar proposals done in the field of satellite links, we consider a more general way of calculating the inter-departure time of ACKs. Moreover, we provide a detailed algorithm that clarifies the operation and the implementation. Finally, we illustrate how a dynamic parameter reconfiguration is capable of maintaining an optimum performance, avoiding buffer overflow and excessive latency in situations of sudden changes in the RB bandwidth.

An important advantage of our algorithm compared to random or RED-like mechanisms is its deterministic operation, which reduces the computational cost of the algorithm, and makes it more feasible for its implementation at the RLC level where the buffering is done in a per-user basis.

Some additional advantages of our algorithm are that it does not require changes in 3G specifications nor in TCP itself, and preserves TCP end-to-end semantics.

ACKNOWLEDGMENTS

This work was supported by the Spanish Inter-Ministerial Sciece and Technology Commission under project TEC2005-08068-C04-01/TCM.

REFERENCES

[1] 3GPP TS 25.322, "Radio Link Control (RLC) protocol specification", v. 6.4.0., Jun. 2005.

[2] A. Gurtov, S. Floyd, "Modeling Wireless Links for Transport Protocols", *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, Apr. 2004, pp. 85-96.

[3] R. Chakravorty, A. Clark and I. Pratt, "Optimizing Web Delivery over Wireless Links: Design, Implementation and Experiencies", *IEEE J. Select. Areas Commun.*, vol. 23, no. 2, Feb. 2005, pp. 402- 416.

[4] Jing Wu et al.,"ACK Delay Control for Improving TCP Throughput over Satellite Links", *Proc. IEEE ICON'99*, Oct. 1999, pp. 303- 312.

[5] M. Rossi et at., "On the UMTS RLC Parameters Setting and their Impact on Higher Layers Performance", in *Proc. IEEE 57th VTC*, vol. 3, Oct. 2003, pp. 1827- 32.

[6] J. J. Alcaraz, F. Cerdan and J. García-Haro, "Optimizing TCP and RLC Interaction in the UMTS Radio Access Network", *IEEE Network*, vol. 20, no. 2, Mar. 2006 pp. 56 - 64.

[7] A. Chockalingam and M. Zorzi, "Wireless TCP Performance with Link Layer FEC/ARQ", in *Proc. IEEE ICC*, June.1999, pp. 1212-16.