Cross-Layer Design to Improve Wireless TCP Performance with Link-Layer Adaptation

Toktam Mahmoodi, Vasilis Friderikos, Oliver Holland, A. Hamid Aghvami
Centre for Telecommunication Research, King's College London,
Strand, London WC2R 2LS, UK
{toktam.mahmoodi, vasilis.friderikos, oliver.holland, hamid.aghvami}@kcl.ac.uk

Abstract—**Transmission Control Protocol (TCP), the almost universally used reliable transport protocol in the Internet, has been engineered to perform well in wired networks where packet loss is mainly due to congestion. TCP throughput, however, degrades over wireless links, which are characterized by a high and greatly varying bit error rate and by intermittent connectivity. Over such wireless links, the performance achieved by TCP can be improved through the use of cross-layer algorithms at the link-level, which interact with the TCP state machine. In this paper, a TCP-aware dynamic ARQ algorithm is therefore proposed, which utilizes TCP timing information to prioritize ARQ packet retransmissions. Numerical investigation of the proposed algorithm demonstrates the performance improvements that can be attained through this approach, in comparison with TCP-agnostic link-layer approaches.**

Keywords–TCP; cross-layer design; wireless networks

I. Introduction

The evolution to wireless networks creates new challenges in communications. Moreover, newer generations of wireless networks aim to support higher data rates and carry heavier traffic while maintaining a low cost. To meet these challenges, cross-layer architectures have become an important topic in improving the performance of wireless networks. This topic is a motivation for numerous researches to examine the issue of cross-layer design principles on wireless networks in addition to answering the main question; If it is leading to a good architectural design or not [1].

Transmission Control Protocol (TCP) is by far the most widely used reliable end-to-end transport protocol supporting congestion control over the Internet. Hence TCP is also well used over wireless networks, as a significant majority of connections over such networks are using the Internet at some point in the end-to-end communication path. TCP reacts to all packet losses as being an indication of congestion somewhere in the end-to-end path. On the other hand, over wireless links losses can be the effect of the higher bit error rates experienced by these links, thus leading to TCP incorrectly attributing these losses to congestion, hence incorrectly reducing its transmission rate. Numerous approaches have therefore been proposed in the literature to optimize TCP performance in wireless networks [2], [3]. These works can be categorized into two main classes: The first class of researches, modify TCP to improve TCP compatibility to wireless networks, and the second class tries to improve TCP performance with no modifications to the TCP state machine but by modifying algorithms in other layers.

This paper focuses on the second category of solutions, which are further described in the Section II. We attempt in this work to achieve an intelligent link-layer through using information obtained by a cross-layer interaction with the TCP transport layer. More specifically, the proposed scheme utilizes the Round Trip Time (RTT) of the TCP flows to prioritize the (re)transmissions so that unnecessary time out events are avoided. A *TCP-aware dynamic* Automatic Repeat reQuest (ARQ) algorithm is proposed, which improves end-to-end performance without modifying the TCP state machine. The main contributions of this paper are therefore the introduction of this dynamic ARQ scheme, and the subsequent investigation of its effects on performance using a realistic TCP model within the OPNET simulation platform.

The remainder of this paper is organized as follows. In the next Section, the second aforementioned category of research on TCP performance optimization is outlined. In Section III, the focus is on ARQ algorithms in the link-layer, where our novel adaptation of the the link-layer ARQ algorithm is also described. In Section IV our simulation results are presented and Section V concludes and discusses potential future work.

II. TCP over Wireless Links

TCP provides a reliable end-to-end transport layer. Most usually, a TCP flow initiates the congestion window (cwnd) to one Maximum Segment Size (MSS), and in the slow start phase doubles it each RTT. In the absence of losses, TCP continues increasing the cwnd until it reaches the Slow Start threshold (SSthresh), whereby it then switches to congestion avoidance phase thence increases it by only 1 MSS per RTT. For each packet loss, dependent on the type of packet loss, the congestion control mechanism of TCP reduces the cwnd in anticipation of congestion [4], by either halving it or by setting it to 1. Due to the time taken for the cwnd to increase again to an optimum rate, this can have the effect of significantly decreasing the throughput if it happens frequently. TCP is therefore designed to perform well in wired networks in which packet losses are usually a result of congestion; however, over wireless links, which suffer from a high bit error rate as well as burst errors due to intermittent connectivity such as hand offs, TCP reacts to all losses as congestion, hence unnecessarily reduces throughput.

Various studies of the impact of retransmissions on throughput have been performed in the literature, commencing with Jacobson's experiments on congested wired networks [5] which led to the improvements incorporated into the TCP Reno protocol [6]. These improvements are particularly embodied as Fast Retransmit and Recovery (FRR)[1]. More recently, extensive efforts have been expended on modifications to improve the performance of TCP via advanced algorithms implemented in other layers of the protocol stack. Such efforts usually attempt to avoid the TCP congestion control mechanism being incorrectly triggered as a result of random wireless errors. Some of them are briefly discussed as follows.

A. Proxy TCP

The Snoop protocol [7] is one of the first such methods. The "Snoop Agent", implemented at the local wireless link (e.g. the base station), works by not passing duplicate acknowledgments to the TCP layer, and instead retransmits the lost packet locally. Through this approach, TCP is not aware of packet losses and the cwnd is therefore not incorrectly reduced.

B. Reliable Link-Layer

Another approach is to use a reliability enhancing methods at the link-layer to prevent packet losses over the wireless channel. ARQ and Forward Error Correction (FEC) can provide this reliability at the local link [8], [9], [10]. The drawback of ARQ is that it causes the RTT to fluctuate, and the drawback of FEC is that, dependent on implementation, it may consume extra bandwidth through pro-actively transmitting redundant information. These drawbacks present the need for careful consideration in the use of link-level ARQ and FEC, based on application type. For example, applications which are sensitive to delay may prefer not to use an ARQ mechanism, and may rely on FEC instead [11]. Hence, the aim of this paper is to find the best reliability approach for the wireless link, in order to enhance end-to-end TCP performance.

C. TCP and ARQ

As discussed, a common approach to improve TCP performance is the use of some form of ARQ mechanism that prevents the TCP source from misinterpreting packet losses as being caused by congestion. TCP over ARQ has therefore been extensively studied in the last few years [12], [13]. As mentioned above, ARQ potentially increases, or at least causes fluctuations in the RTT of TCP, and this may interfere with the TCP timeout. The TCP retransmission timer may expire at the time a lost packet is being retransmitted over the wireless link. Such shortcomings, can be improved by adapting the maximum number of local link retransmissions dependent on the TCP state machine. Reference [14] therefore attempts to optimize the number of link-layer retransmissions dependent on the end-to-end packet loss rate perceived by TCP.

[1]If the sender receives three duplicate acknowledgements, it assumes that the data segment indicated by the acknowledgements is lost and immediately retransmits the lost segment. With FRR, time is not lost waiting for a timeout in order for a retransmission to begin.

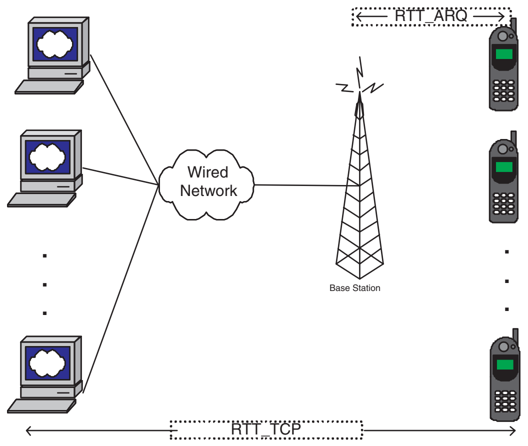

Figure 1. RTTs at the end-to-end TCP layer and at the local link-layer

III. Link-Layer Adaptivity

Retransmissions at the link-layer are performed via the local link ARQ algorithm, which has two main advantages compared with TCP end-to-end retransmissions. Firstly, retransmissions at the link-layer are performed faster, due to the local link having a much smaller RTT– this is seen in Fig. 1. Secondly, if retransmissions at the link-layer are successful, packet losses can be hidden from TCP. Thus the cwnd will not be incorrectly altered.

In this work, we assume the stop-and-wait ARQ algorithm over the link-layer. Unacknowledged packets are therefore successfully retransmitted if the required number of retransmission attempts is less than or equal to the maximum allowed number of retransmissions. We try to select an appropriate value for the maximum number of retransmissions (N_{ret}), as well as an appropriate order of retransmissions, so as to optimize the ARQ algorithm.

A. TCP-Aware Dynamic ARQ

ARQ serves similar reliability functions to TCP, albeit at a different layer. Hence the assignment of ARQ parameters without having taking advantage of information from the TCP state machine is suboptimal for TCP flows. We therefore define a cross-layer interaction with the TCP layer, by assigning N_{ret}, as well as the retransmission priority for each packet, dynamically. An ultimate objective is to achieve a TCP-aware dynamic ARQ algorithm, under the assumption that we possess information on the TCP timer when the Retransmission Time Out ($\mathrm{RTO}_{\mathrm{TCP}}$) is close to expiration. Note that ensuring that the packet arrives rapidly at the receiver becomes more critical in this case. The details of our algorithm are described as follows.

Every non-ACK packet is categorized in a priority queue, according to its weight. The packet weight is assigned based on $\mathrm{RTO}_{\mathrm{TCP}}$ and the number of transmission attempts of the packet. If we denote P_{ij} as the jth packet in the queue from TCP flow i, then w_{ij}, the weighting of P_{ij}, is described by

$$w_{ij} = 10^4 * n_{ij} + RTO_{ij}, \tag{1}$$

where n_{ij} is the number of reattempts of packet P_{ij}. The RTT of TCP is of the order of one second (10^3 ms), thus the coefficient 10^4 normalizes the weighting function value to the RTO_{TCP}.

We subsequently define three queues, namely Q1, Q2, and Q3, and each P_{ij} is placed in one of these queues at a position according to its weight. Retransmission starts from Q1 and is followed by Q2, and then Q3. The main idea for implementing these queues is to avoid retransmitting any specific packet iteratively.

The priority of each packet is changed after each retransmission, by replacing its queue. One main weak point remains in this procedure. If any of the P_{ij}s which are located in Q2 or Q3 have an RTO_{TCP} close to expiration, the RTO_{TCP} will expire before the packet has one more retransmission chance at the local link. Hence to solve this, the retransmission starts from $P_{i_{11}j_{11}}$, and simultaneously $RTO_{i_{21}j_{21}}$ and $RTO_{i_{31}j_{31}}$ are compared with RTT. If either of these is less than 2·RTT, the corresponding packet will be retransmitted first, then the procedure will continue with $P_{i_{11}j_{11}}$. At the beginning of each retransmission, this procedure will be repeated. After the Q1 retransmission is performed, the algorithm will continue retransmissions from $P_{i_{21}j_{21}}$ with the same routine, then retransmits $P_{i_{31}j_{31}}$ etc. The maximum number of retransmission attempts is limited to 3, which is well used number for this purpose in wireless networks. Hence, after the third retransmission attempt, the packet is discarded.

Algorithm 1 presents the complete description of the procedure.

Algorithm 1 TCP-AWARE DYNAMIC ARQ ALGORITHM DESCRIPTION

```
1:  $IF\ w_{ij}\ IS\ LESS\ THAN\ 10^4\ THEN$
        Q1 $\leftarrow P_{ij}$
2:  $ELSE\ IF\ w_{ij}\ IS\ LESS\ THAN\ 2*10^4\ THEN$
        Q2 $\leftarrow P_{ij}$
3:  $ELSE\ IF\ w_{ij}\ IS\ LESS\ THAN\ 3*10^4\ THEN$
        Q3 $\leftarrow P_{ij}$
4:  ELSE DROP THE PACKET
5:  Sort Q1: $RTO_{i_{11}j_{11}} < RTO_{i_{12}j_{12}} < ... < RTO_{i_{1n}j_{1n}}$
6:  Sort Q2: $RTO_{i_{21}j_{21}} < RTO_{i_{22}j_{22}} < ... < RTO_{i_{2n}j_{2n}}$
7:  Sort Q3: $RTO_{i_{31}j_{31}} < RTO_{i_{32}j_{32}} < ... < RTO_{i_{3n}j_{3n}}$
8:  Starts Retransmission from Q1
9:  $IF\ RTO_{i_{11}j_{11}}\ OR\ RTO_{i_{21}j_{21}}\ OR\ RTO_{i_{31}j_{31}}\ IS\ LESS\ THAN\ 2*RTT\ THEN$
        Retransmit the corresponding packet
    $ELSE$
        Retransmit $P_{i_{11}j_{11}}$
10: Continue with Q2 and then Q3
```

Another issue that needs to be addressed is the RTT measurement. The RTT is estimated passively, by using the Timestamp option or the TCP self clocking method. The Timestamp option regarding RFC 1323 [15], is placed in the TCP header. TCP headers containing this option will increase from 20 bytes in size to 32 bytes. The receiver then echoes the Timestamp value in the acknowledgement, thus allowing the sender to calculate an RTT for each received ACK.

The self-clocking RTT estimation method does not have to rely on TCP Timestamps. Hence it can be used in the absence of TCP Timestamp option. In [16] two methods are proposed to passively measure the RTT at an intermediate measurement point, which have an error rate of less than 10%. Using these methods, the achieved accuracy is related to the measurement location, which must be not too far from the sender. Another technique, proposed in [17], measures RTT passively in the middle of the path with an error rate of less than 10% for 90% of the senders.

Given the estimated RTT, the RTO is calculated as [4]

$$RTO = RTT + 4 * D_{RTT}, \qquad (2)$$

where D_{RTT} is the deviation of the RTT.

B. Cross-Layer Point of View

The objective is for us to optimize our cross-layer design with the lowest Complexity, while keeping the existing protocol stack and aspiring to longevity and stability in the design. To achieve all of the above, a cross-layer information box is created for the interaction between TCP and the link-layer, in order to transfer information to our TCP-aware dynamic ARQ algorithm. More detail on the implementation model is given in the following Section.

IV. Simulation

To observe how the TCP performance is influenced by our TCP-aware dynamic ARQ approach, a Wireless Local Access Network (WLAN) model within the OPNET platform is simulated. The TCP state diagram implementation in OPNET is the same as described in RFC 793 [18]. TCP Reno is assumed in this work, as it is a commonly-used transport protocol in the Internet. The aforementioned Timestamp option is enabled, which is incremented by one every 500ms. This is well within the recommendation to increment by one at an interval of between 1ms and 1s. Link-level retransmissions are performed using the stop-and-wait ARQ algorithm.

In the considered scenarios, the WLAN nodes setup connections with wired servers via a single WLAN Access Point (AP) and wired routers. This is a similar configuration to that depicted in Fig. 1. Our TCP-aware dynamic ARQ functionality is placed in the AP. A cross-layer information box is placed in the protocol stack of the AP, in order to extract TCP timing information and transfer it to our algorithm implemented in the link-layer. Weights are assigned to packets according to the system described in Section III.A, and this information is passed to the ARQ algorithm.

Specific simulation parameters are given as follows: -

- Simulation Duration: 600s,
- FTP servers: 16MB file download size,
- HTTP servers: HTTP1.1, page interarrival time = exponential (mean 60s), object size = exponential (mean 5kB), number of objects per page = exponential (mean 7),

- Email servers: Email size: 2kB, Interarrival time = exponential (mean 120s),
- TCP Maximum Segment Size (MSS): 1460B, Reno TCP,
- MAC buffer size: 32kB,
- MAC frame size: 320B (Fragmentation enabled),
- Physical-Layer characteristic: OFDM (802.11a, 6Mbps),
- Operating frequency: 5.4GHz.

The channel is modeled with the path loss ($\alpha = 3$), Rayleigh fading (exponential random variable, $\beta = 1$) and LogNormal shadowing (standard deviation 4dB). In each scenario, every mobile terminal connects to a unique server via the wireless AP and the wired routers. The mix of traffic includes FTP, Web browsing and Email traffic, where in all scenarios, 40% of the clients are FTP users, 35% are HTTP users and the other 25% are Email users. Simulation results are presented in terms of end-to-end throughput, comparing our dynamic ARQ scheme with the normal 802.11a scheme, using a maximum of 3 retransmission attempts in both cases. The maximum value of the aggregated end-to-end throughput shows an approximate $15-60\%$ improvement, in various conditions, for our dynamic ARQ scheme as compared to the default scheme.

In the first simulated scenario using 15 wireless clients, the RTT for each flow is a uniformly distributed random variable with bounds set 5ms and 100ms. Throughput for this scenario, versus wireless link distance, is plotted in Fig. 2. Results here, show that the dynamic ARQ scheme improves the end-to-end throughput for a range of mobile clients being connected to the AP.

In a second simulated scenario, 15 wireless clients are mobile at a fixed distance of 55m from the AP. RTTs of end-to-end paths are set according to different random distributions: Uniform (a = 5ms, b = 100ms), Normal (μ = 50ms, σ = 20ms), and Exponential ($\beta = 50ms$). Results for this scenario, presented in Table 1, show that the throughput improvements of our dynamic ARQ scheme are largely unaffected by the specific RTT distribution.

In a third simulation scenario, using the same uniform distribution for end-to-end paths' RTTs, wireless clients are in the distance of $55m$ from the AP. Throughput for this scenario is plotted against the number of clients in the simulation, varied from 5 through 20, in Fig. 3. The results in Fig. 3. show the scalability of our dynamic ARQ scheme, which still levies a significant performance improvement even if there is a high numbers of clients.

In the fourth simulated scenario, RTTs are again given the same uniform random variable, and wireless clients are placed a uniformly distributed distance from the AP (a = 40m, b = 70m). Results for this scenario show a 32% improvement in end-to-end throughput.

In addition to throughput improvement, it can be seen that TCP retransmissions is decreased using our scheme. These results are summarized in Table 2. for the above 4 simulated scenarios.

In the fifth and final simulated scenario using 15 wireless clients, and the same RTTs for the end-to-end paths, the wireless link Bit Error Rate (BER) is increased from 10^{-6} to 10^{-3}. Throughput for this scenario is plotted in Fig. 4, which shows the performance improvement increases as the BER of the channel goes higher.

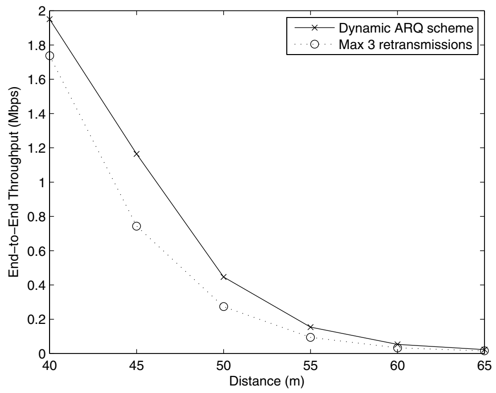

Figure 2. Max. end-to-end aggregated throughput vs. wireless propagation distance for the first simulated scenario

With regard to simulation results presented in Tables 1-2 and Figs. 2-4, the proposed TCP-aware dynamic ARQ algorithm improves TCP performance in the variety of scenarios. Packet loss experienced by TCP is decreased, and end-to-end throughput is increased. Additionally, our TCP-aware dynamic ARQ approach presents a TCP-friendly link-layer. Some other advantageous characteristics of the proposed scheme can be summarized as follows: -

- It performs RTO-based link-level retransmissions, which makes the link-layer compatible with the TCP limitations.
- The algorithm is of low complexity from both the processing and buffering point of view. The highest complexity procedure is sorting packets with regard to the weighting function–this complexity is O(log n). The scheme does not affect queuing or storage memory requirements of the link-layer.
- Using the priority queuing idea as presented, no packet will remain at the same level of priority after it's retransmission. Therefore, the same packet will not be retransmitted continuously.
- The algorithm allows packets which are close to expiration to jump the queue and to get a last retransmission chance within their RTO_{TCP}.
- Our dynamic ARQ scheme avoids wasted retransmissions being sent after the associated packet's RTO_{TCP} timer has expired.

V. Conclusion and Future Works

In this paper, we have investigated the effect of the maximum allowed number of retransmissions (N_{ret}) of the stop-and-wait ARQ algorithm on TCP performance. This has been done utilizing the OPNET simulation platform. A TCP-aware dynamic ARQ algorithm has been proposed, which dynamically adapts the maximum allowed number and priority of transmission reattempts, based on two parameters: The TCP

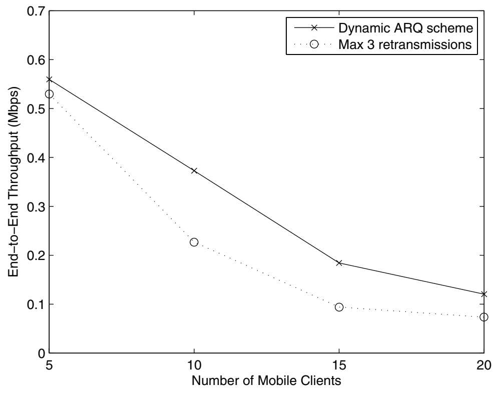

Figure 3. Max. end-to-end aggregated throughput vs. number of wireless clients for the third simulation scenario, propagation distance is 55m

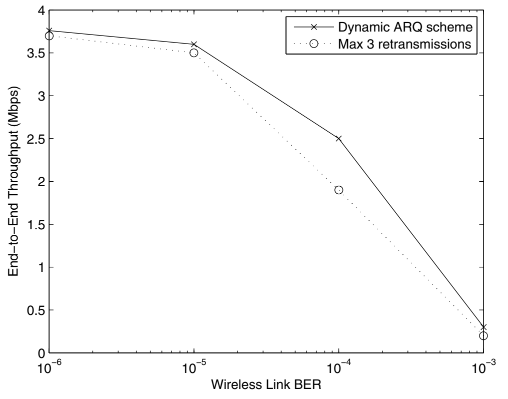

Figure 4. Max. end-to-end aggregated throughput vs. wireless link BER

retransmission time-out, and the number of retransmissions which have thus-far been sent. A devised packet priority weighting function assists the retransmission process by receiving information via an information box implemented as the cross-layer interaction between TCP and the link-layer. Specifically, this box extracts information on TCP retransmission time-outs, and passes it to the aforementioned algorithm which operates at the link-layer. The results presented in Tables 1-2 and Figs. 2-4 show a $15 - 60\%$ improvement in end-to-end performance through our novel approach.

In future work, it would be interesting to further investigate the influence on end-to-end performance of including extra TCP parameters, such as the cwnd, in the packet priority weighting function.

VI. Acknowledgment

The work reported in this paper has formed part of the Delivery Efficiency Core Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com. This research has been funded by EPSRC and by the Industrial Companies who are Members of Mobile VCE. Fully detailed technical reports on this research are available to Industrial Members of Mobile VCE.

TABLE I
Average percentage of TCP retransmitted packets in the first, second, third and fourth scenarios.

Simulated Scenario	1^{st}	2^{nd}	3^{rd}	4^{th}
Dynamic Scheme (ret. packets)	10%	9.7%	11.5%	10.4%
3. Ret Scheme (ret. packets)	11.6%	11.2%	13%	12%

TABLE II
Maximum end-to-end aggregated throughput for the Uniform, Normal and Exponentially distributed RTTs

RTT Distribution	Uniform	Normal	Exponential
Throughput (kbps): Dynamic	550	560	325
Throughput (kbps): 3. Ret	280	340	275

References

[1] V. Kawadia and P. R. Kumar, “A Cautionary Perspective on Cross-Layer Design,” *IEEE Wireless Comm.*, vol. 12, pp. 3–11, Feb. 2005.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links,” *IEEE/ACM Trans. Net.*, vol. 5, pp. 756–769, Dec. 1997.

[3] V. Tsaoussidis and I. Matta, “Open Issues on TCP for Mobile Computing,” *Wireless Comm. and Mobile Comp.*, vol. 2, pp. 3–20, Feb. 2002.

[4] W. R. Stevens, *TCP/IP illustrated, Volume I The protocols*. Addison Wesley, Feb. 2000.

[5] V. Jacobson, “Congestion Avoidance and Control,” *Proc. ACM SIGCOMM ’88*, Stanford, CA, USA, Aug. 1988.

[6] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” *IETF RFC 2581*, Apr. 1999.

[7] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, “Improving TCP/IP Performance over Wireless Networks,” *Proc. ACM MOBICOM ’95*, Berkeley, CA. ACM, USA, Nov. 1995.

[8] M. Assaad and D. Zeghlache, “Cross-Layer Design in HSDPA System to Reduce the TCP Effect,” *IEEE JSAC*, vol. 24, pp. 614–625, Mar. 2006.

[9] F. Vacirca, A. De Vebductis, and Baiocchi, “Optimal Design of Hybrid FEC/ARQ Schemes for TCP over Wireless Links with Rayleigh Fading,” *IEEE Trans. Mobile Comp.*, vol. 5, pp. 289–302, Apr. 2006.

[10] J. J. Alcaraz, F. Cerdan, and J. Garca-Haro, “Optimizing TCP and RLC Interaction in the UMTS Radio Access Network,” *IEEE Net.*, vol. 20, pp. 56–64, Mar./Apr. 2006.

[11] R. Abdelmoumen, M. Malli, and C. Barakat, “Analysis of TCP Latency over Wireless Links supporting FEC/ARQ-SR for Error Recovery,” *Proc. ICC ’04*, vol. 7, pp. 3994 – 3998, Paris, France, Jun. 2004.

[12] A.-F. Canton and T. Chahed, “End-to-End Reliability in UMTS: TCP over ARQ,” *Proc. IEEE GLOBECOM ’01*, vol. 6, Texas, Nov. 2001.

[13] A. L. Toledo and X. Wang, “TCP Performance over Wireless MIMO Channels with ARQ and Packet Combining,” *IEEE Trans. Mobile Comp.*, vol. 5, pp. 208– 223, Mar. 2006.

[14] F. Vacirca, A. D. Vendictis, A. Todini, and A. Baiocchi, “On the Effects of ARQ Mechanisms on TCP Performance in Wireless Environments,” *Proc. IEEE GLOBECOM ’03*, vol. 2, pp. 671–675, CA, Dec. 2003.

[15] V. Jacobson, R. Braden, and D. Borman, “TCP Extensions for High Performance,” *RFC 1323*, May 1992.

[16] B. Veal, K. Li, and D. Lowenthal, “New Methods for Passive Estimation of TCP Round Trip Times,” *Lecture notes in Computer Science*, vol. 3431, pp. 121–134, 2005.

[17] S. Jaiswal, *Measurement in the Middle: Inferring End-End Path Properties and Characteristics of TCP Connections through Passive Measurement*. PhD thesis, University of Massachusetts Amherst, Sep. 2005.

[18] “TCP Model User Guide,” *OPNET Online Documentation, Rel 11.5*, http://www.opnet.com.