Efficient Internet Traffic Delivery over Wireless Networks

Rami G. Mukhtar, Stephen V. Hanly and Lachlan L. H. Andrew

June 16, 2003

Abstract

The high demand for wireless Internet connectivity has driven the development of highly efficient radio link technologies. However, their performance can be compromised by inadvertent interactions with the higher layer TCP flow control protocol. If we are to maximize the performance of wireless links, then we must ensure that mechanisms operating at every layer of the protocol stack interact efficiently. In this article we provide a brief tutorial of some of these radio link enhancements. We then outline how higher layer flow control protocols should behave, and outline techniques for taming the behavior of TCP, to ensure that the performance of lower layer enhancements are not compromised.

1 Introduction

Randomness is an inherent characteristic of wireless communications. It is a feature that stems from a combination of physical phenomena, including specular reflections, and multiple radio propagation paths to the receiver. Due to the mobility of users and/or other objects in their vicinity, these phenomena will induce time variations in the quality of the channel between transceivers, which is often referred to as fading. A great deal of wireless research is focussed on mitigating these effects. Indeed, enormous progress has been achieved over the past 50 years in this area, exploiting techniques from coding, signal processing and information theory. Modern wireless cellular telephony is built upon these advances.

The demand for wireless Internet access is growing, and new performance requirements are emerging. Distinct from the fixed bandwidth and latency requirements of traditional voice traffic, the majority of Internet traffic is *elastic*. Elastic traffic is a broad class that encompasses many of the most popular applications used on the Internet today, including the World

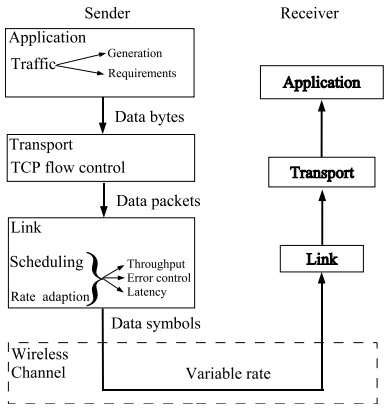

Figure 1: Overview of the Internet Stack Architecture.

Wide Web (WWW), File Transfer Protocol (FTP), Peer-to-Peer clients and Electronic Mail. In contrast to voice traffic, there is a paramount objective for elastic traffic: minimization of the total file transmission time. For many applications, including the WWW, latency is still an important issue, but there is no longer a hard constraint on latency, as there was for voice.

The hardware, software and protocols that facilitate data transmission over the Internet can be categorized into distinct layers, as depicted in Figure 1. Data traffic generated by applications is passed to the transport layer. In the case of the Internet, this is most often the Transmission Control Protocol (TCP). TCP is an end-to-end protocol that controls the rate at which packets from a given source are injected into the network (i.e. flow control), with the objective of maximizing the performance of the network. The link layer multiplexes multiple sources over a single link, to deliver data bits efficiently and reliably over the physical channel.

The elastic nature of the majority of Internet traffic has motivated the development of adaptive wireless transmission techniques, which are usually implemented at the link layer. These techniques obtain a diversity gain by exploiting variations of the wireless channel quality both over time and over the user population. Total overall throughput is increased, at the cost of variable packet transmission latency. The user perceived performance of these mechanisms is highly dependent on a combination of the application requirements and the behavior of other mechanisms operating in different layers of the Internet protocol stack. In essence, designing an efficient wireless network is a multi-layer discipline.

This article provides a brief tutorial of some of the most recent and popular of these link layer enhancements. It is then seen in Section 3 that the performance of these optimizations can be jeopardized by the behavior of flow control mechanisms operating at the transport layer, such as the TCP. One popular solution to this problem is receiver side flow control, which is

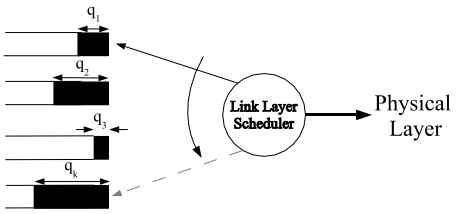

Figure 2: Link Layer Scheduler with per user queueing.

discussed in Section 4. It is demonstrated how an algorithm suitable for this type of deployment can be used to tame the behavior of TCP. This ensures that it does not interfere with the performance of these lower layer optimizations, and enhances the performance perceived by the user.

2 Link Layer Optimizations

2.1 Packet Transmission Scheduling

Packet scheduling can considerably increase throughput by exploiting the temporal fluctuations in users' channel qualities. Consider the down link mode of operation, where a mobile user is receiving data from a central base station. Typically a base station will simultaneously service a number of users. At any one time instant some users will enjoy better channel conditions than others. In fact, it has been shown that the performance of wireless systems can be enhanced by using multiple antennas to artificially induce temporal fluctuations in channel quality [1]. If a base station is limited to transmitting data packets in a particular order, for example in the order that they reside in a queue, it will be unable to capitalize on the temporal diversity of the users' channel qualities. This has led to proposals that segregate users' packets into separate buffers, providing the opportunity for a scheduler to select the optimal user to transmit to at an instant of time [2, 3, 4].

Figure 2 is a schematic depiction of a packet scheduler for k users. Each user, i, has a queue of packets awaiting transmission, of length $q_i(t)$. Typically, a scheduler will decide the next queue to be serviced based on a combination of measurements of the users' current signal to noise ratios, a measure of the users' mean channel rates (calculated over some time interval), the users' priorities, and the current size of the users' queues. For example, if a user temporarily suffers reduced channel quality, then the scheduler might serve that user less frequently, giving preference to users with a higher channel quality. When the user enjoys better channel conditions, the channel rate to that user will then be increased, making up for the reduced capacity

earlier. These schemes count on temporal fluctuations in channel quality.

The simplest channel-aware scheduling policy selects the user with the best channel quality. Although this scheduler has been shown to achieve the maximum possible aggregate rate [2], it can lead to throughput starvation for some users, and long term unfairness. To remedy this problem, practical schedulers often take into account the average throughput a user receives [3]. Users that suffer from a lower long term average throughput are preferentially scheduled. This leads to better long term fairness amongst the users, at a cost of reduced overall capacity gain. Another approach is to deem the base station's transmission time (as opposed to throughput) to be the resource that must be apportioned. Under this scheme, in the long term, users are given an equal share of the base station's transmission time. The base station makes its scheduling decision based on the users' instantaneous channel qualities and the long term proportions of time that they receive data [4].

Scheduling decisions can only be made optimally if each of the queues has packets queued, which are awaiting transmission. Emptying of a user's packet buffer at the base station, due to failures of the transport layer flow control to maintain a supply of packets, can lead to suboptimal scheduling. Thus, it is important that the transport layer flow control mechanism does not inadvertently cause a user's buffer to drain, by unnecessarily holding packets up at the source.

2.2 Link Layer Rate Adaptation

Current wireless standards ensure reliable delivery by using link layer mechanisms to adapt their transmit rate to the channel quality. Nanda *et al.* [5] provide an overview of current link layer techniques, and how they are applied in different cellular standards today. Incremental Redundancy (IR) techniques effectively vary the code rate on a transmission slot by slot basis, tracking fluctuations in the channel quality. Link Adaptation (LA) techniques measure the average channel quality (over several transmission slot intervals) and choose an appropriate modulation scheme and/or coding scheme. Hence, IR provides a mechanism for faster rate adaptation than LA.

The net effect of these interacting rate adaptation mechanisms is that the time series of successful packet transmission times, as perceived by a transport layer flow control scheme, will be random and non-stationary. Its behavior will vary on two distinct time scales. Statistics, such as the mean transmission rate, will vary on a timescale much slower than a packet transmission time, according to the choice of LA and packet scheduling policy. On a much faster time scale, packet transmission times will themselves be random, as a result of link layer retransmissions and IR rate adaptation.

3 Transport Layer Objectives

3.1 TCP: An Overview

TCP is the Internet's most popular transport layer delivery service. It provides reliable data delivery to the application layer. TCP automatically organizes bytes into packets of an appropriate size and then ensures that they are reliably delivered to their destination. TCP boasts several features, including automatic packet retransmission and reordering, transmission error detection, appropriate packet size discovery, and flow control. The last feature, flow control, is particularly relevant to this article. TCP's flow control algorithm ultimately determines how much data is buffered in the network, in-flight between the source and destination. As highlighted in Section 2, this is important in determining whether lower layer optimizations can operate effectively. Before further elaborating on what would be desirable from a transport layer flow control system, let us briefly examine some pertinent features of TCP's flow control algorithm.

TCP is a window based flow control algorithm. That means that the number of unacknowledged bytes that are sent into the network is limited to at most a particular *window size*. The TCP window size is set to be the minimum of the currently computed congestion window (CWND), which is set by the sender, and the receiver's advertised window (AWND), as advertised by the receiver. The purpose behind AWND will be explained later. At the sender, the TCP flow control algorithm attempts to mitigate congestion within the network by controlling the value of CWND. The control algorithm is in fact a combination of several distinct algorithms, which are activated at various stages of a TCP connection's lifetime. The two dominant algorithms are: *slow start* and *congestion avoidance*. The slow start algorithm increases the value of CWND at an exponential rate, doubling the window size every round trip time. Starting at one packet, it usually terminates when it fills the network pipe and a packet loss occurs. The termination of the slow start algorithm is followed by congestion avoidance. Congestion avoidance increases the value of CWND linearly, at a rate of approximately one packet per round trip time, probing for any spare capacity. Eventually, a buffer overflow will result, and a packet loss will occur. This prompts TCP to halve its window size. In the simplest sense, congestion avoidance is a form of the Additive Increase Multiplicative Decrease algorithm (AIMD) [6], with the decrease factor set to 0.5. Under normal operation, congestion avoidance should dominate the evolution of the value of CWND. This causes a *sawtooth* evolution of the window size, which is all too familiar to people who have worked with the protocol. See Figure 3 for an illustration of this behavior.

Let us consider the case when there is a single bottleneck link for a

TCP connection, as is usually the case for a variety of wireless and wireline access network technologies. A wireline link's transmission rate is usually constant. In this case, high utilization can be achieved by dimensioning the bottleneck buffer to be larger than the product of the bottleneck bandwidth and total delay in the rest of the network. This will cause TCP's window size to oscillate between one and two times the bandwidth delay product, which prevents the link from being starved. This works quite well for wire line networks, although it should be noted that queueing requirements at the bottleneck link is of the order of the bandwidth delay product, which can be very large for high capacity links that span oceans. Nevertheless, throughput need not be compromised.

In a wireless network the situation is even worse. As discussed in Section 2.2, the link rate is now a random quantity, which varies on several distinct time scales. In order to keep the link busy, we could compensate for these fluctuations and the sawtooth evolution of the window size by setting the link buffer to be very large. However, TCP's congestion avoidance algorithm will attempt to fill this buffer. If the link rate drops suddenly, then excessively large transmission latencies will result. In contrast, a small buffer would cause the link to be frequently starved of packets, interfering with lower layer optimizations.

3.2 Supporting a Wireless Link Layer

Having exposed the weaknesses of TCP's congestion avoidance algorithm, we now address the question: "How should the transport layer flow control behave to ensure lower layer mechanisms can function optimally?"

A transport layer control algorithm is incapable of responding to random fluctuations in rate, which occur on a time scale smaller than the Round Trip Time (RTT). This is because it takes at least one RTT before the sender can be informed of a change in order to respond. However, transport layer control schemes can respond to mean changes in rate that occur over a longer time period (of the order of a RTT). The objective of a transport layer control system should be to ensure that it does not interfere with the performance of the underlying link layer scheduling mechanisms, whilst automatically adapting to long term changes (several RTTs) in packet transmission times.

To achieve this, each user should maintain a queue at the base station that empties only infrequently, to ensure a steady supply of packets, so that fast link layer rate adaptation mechanisms can take advantage of the fast fluctuations in channel rate. In contrast, slow changes in the mean link rate and other link rate statistics, as a result of the slow link layer rate adaptation mechanisms discussed above, will require the transport layer control to adapt its transmission window size to maintain a mean target queue size. The result is quite distinct from the cyclic fluctuations in queue size that results from

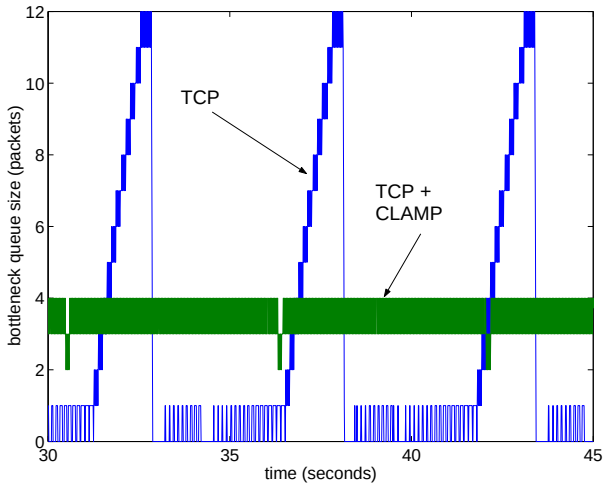

Figure 3: Bottleneck link queue size, with and without CLAMP receiver side control.

TCP's congestion avoidance algorithm. TCP may periodically starve the scheduler of packets, limiting its ability to capitalize on temporal variations in channel quality. See Figure 3 for an illustration of these two distinct behaviors.

Furthermore, a single mobile terminal can have multiple simultaneous TCP connections. In spite of all the advances in radio technology, wireless bandwidth will always be a precious resource, and for this reason, users will often want to differentiate between flows, prioritizing some flows to obtain more of the available capacity than others. For instance, this enables the automatic suspension of a running FTP session when the user reactivates a WWW session by clicking on a link.

Maintaining a constant average queue size and flow differentiation are the two key objectives of flow control over wireless networks.

4 Explicit Window Adaptation

Most TCP performance enhancements for wireless [7] attempt to improve performance by obscuring events that TCP would otherwise inadvertently perceive as indications of congestion. These include packet loss and extraneous packet delays due to repeated link layer retransmissions. However, none of these techniques fundamentally change the behavior of TCP to satisfy the objectives outlined in Section 3. This immediately poses the question: How can we fundamentally change the behavior of TCP without making any modification to the senders or widespread changes to the Internet? Fortunately, we can meet this demand by exploiting a legacy component that

was incorporated into the TCP protocol at its inception, the AWND.

The original function of the AWND was to enable the receiver to limit the sender's window size to avoid running out of computer memory. This was a significant problem in the early days of computer communications as computer memory was expensive, and processor speeds were relatively slow. In present times the AWND value serves little purpose. High processor speeds, and large computer memories, rarely allow it to deviate far from its maximum value. By actively controlling AWND at the receiver, this legacy TCP feature provides the receiver with control over the TCP sender's behavior. This technique is often referred to as receiver side explicit window adaptation [8]. It is compatible with all TCP senders. It only requires that modifications be made to either the receiver, or an intermediate node, which intercepts acknowledgements sent by the receiver, and overwrites the advertised AWND value with a newly computed AWND value. Recall that TCP's window is the minimum of CWND and AWND. If AWND is carefully controlled to ensure that packet loss does not result, then the congestion avoidance algorithm will monotonically increase CWND. In this situation, full control of the window size is available via the AWND mechanism. However, if a packet loss were to occur, then CWND would be decreased. Under these circumstances, there may be periods when the value of CWND dominates the value of the window size. Ideally, for receiver side explicit window adaptation to be successful, it is important to control AWND such that packet loss due to buffer overflows are kept to a minimum.

By definition, receiver side control operates at the receiver. Thus the information available to any algorithm that is based on receiver side control is limited. For instance, measuring the round trip time would require sending explicit probe packets to the sender. Additionally, the sending application may pause transmission at any time during the lifetime of an active TCP session. For these reasons, a receiver side control mechanism should not rely on any direct knowledge of round trip times or the number of active flows over the wireless link.

4.1 An Algorithm

CLAMP [9] is a recently proposed solution for receiver side explicit window adaptation that satisfies the objectives outlined in Section 3. The operation of CLAMP is illustrated schematically in Figure 4. A software agent located in the wireless base station monitors the transmit queue, q, and computes the value of a *price* function, $p(q)$, which is shown in Figure 5. The price value is then inserted into the option field of the TCP header of each outgoing packet.

For each packet received by the client, the price value is read from the TCP header. It is combined with a priority factor, τ, and an estimate of the

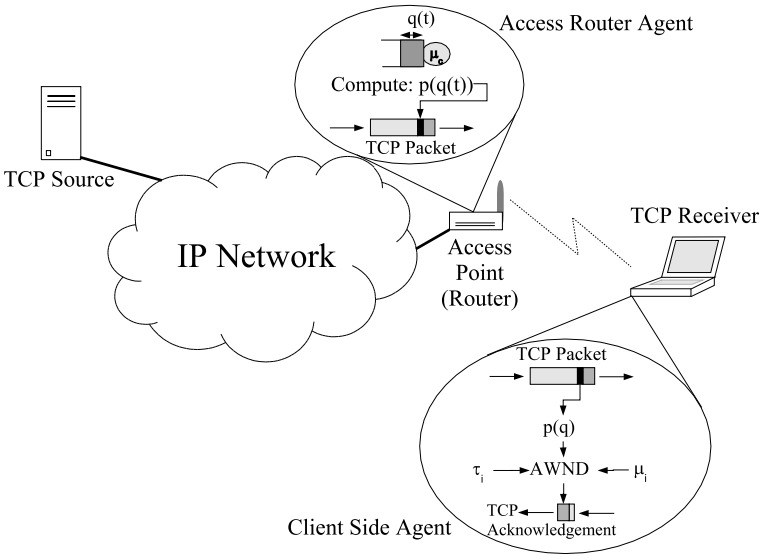

Figure 4: A typical deployment of the CLAMP system.

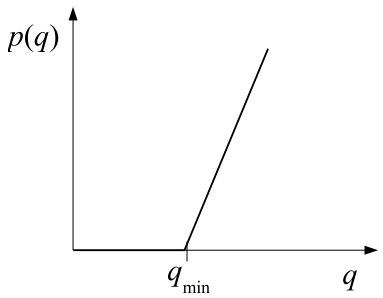

Figure 5: CLAMP pricing function.

current receive rate, μ, to produce a new AWND value that is sent to the sender in outgoing acknowledgements. The update rule is given by

$$AWND_{\text{new}} \leftarrow AWND_{\text{old}} + \frac{(\tau - p(q)\mu)\,\Delta}{d},$$

where Δ is the time elapsed since the last acknowledgment was sent, and d is AWND$/\mu$ averaged over a long time interval. By setting the value of τ for each flow, the user is able to assign a priority to different flows. A larger value of τ will ensure that a flow obtains a greater proportion of the total obtained rate. The proportion of the total rate obtained by a flow, will be its value of τ divided by the sum of all of the τ's for all flows currently sending data [9]. Note that in practice, the decrease in AWND should not be greater than the received packet size, in order to meet the TCP specifications.

The queue size can be controlled by adjusting the $p(q)$ function. Referring to Figure 5, moving the point $q_{\min}$ determines a lower bound on the mean queue size. The slope of the line determines the sensitivity of the mean queue size to the number of active flows. The parameters of this function can either be set statically to suit certain operating conditions, or they can be dynamically optimized to maximize the performance of lower layer mechanisms.

4.2 Performance Improvement

CLAMP mitigates the effect of TCP's cyclic probing for available capacity and eliminates the periodic halving of the window size that results. This reduces the time that the link is left idle because TCP is holding packets back at the source.

The advantage of mitigating TCP's sawtooth capacity probing can be seen in Figure 6. This shows the tradeoff between delay and throughput for a single TCP flow operating over a network that included a bottleneck link, with exponentially distributed random packet transmission times of mean rate 1.5Mbit/s, and a total end-to-end propagation delay of 0.2s in each direction. For a given buffer size, TCP provides no means of controlling the delay, and so the results for pure TCP show the impact of increasing the buffer size of the bottleneck link. Here, the average *goodput* is the amount of data successfully received, and does not include duplicate packets that were inadvertently retransmitted by TCP. Receiver side control decouples the mean queue size from the buffer size. This allows low delay to be achieved with a low probability of buffer overflow, enhancing the performance of services which do not employ retransmission, which includes most real-time services. Under TCP, low delays can only be achieved by reducing the buffer capacity. This increases the probability of packet loss, and leads to a reduction in the performance of real time services. The results clearly

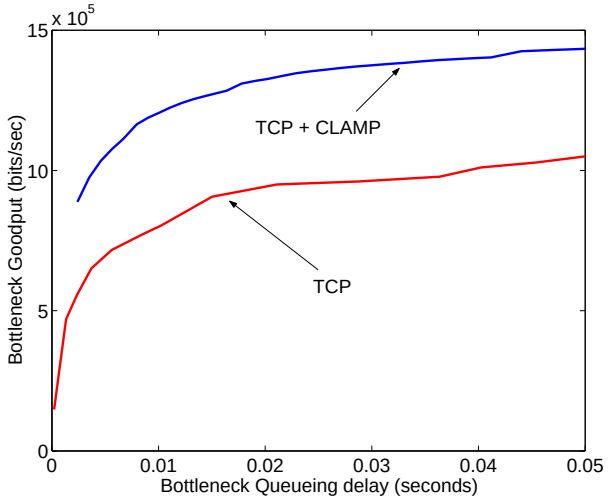

Figure 6: Base station idle time vs. queue size.

demonstrate the increase in throughput and reduction in queueing achieved when receiver side flow control is employed. By mitigating the sawtooth evolution of TCP's window size, a much higher goodput can be obtained for a smaller queueing delay.

Figure 7 shows an example of rate differentiation. This particular example uses CLAMP with $\tau_{\text{www}} = 10$ and $\tau_{\text{ftp}} = 1$. It shows preference being given to the WWW transfer allocating approximately $10/11$ of the capacity to the WWW flow and the remaining $1/11$ to the FTP flow.

5 Conclusion

The elastic nature of the majority of Internet traffic facilitates adaptive wireless transmission techniques that exploit temporal fluctuations in channel quality. However, in order to support these enhancements, it is important to ensure that higher layer flow control mechanisms do not inadvertently interfere with these lower layer mechanisms. In particular, a mean queue of packets at the base station should be maintained, which is just large enough to account for the random fluctuations in packet transmission times, but not so large as to cause an excessive increase in latency.

Receiver side flow control exploits a legacy feature in TCP. It can be used to control existing TCP sources to reach the desired objectives and increase the performance of wireless access to the Internet. Algorithms for receiver side flow control, such as CLAMP, supplement TCP's flow control mechanism. They can be used to ensure that lower layer wireless scheduling mechanisms can perform as intended. Furthermore, they provide users

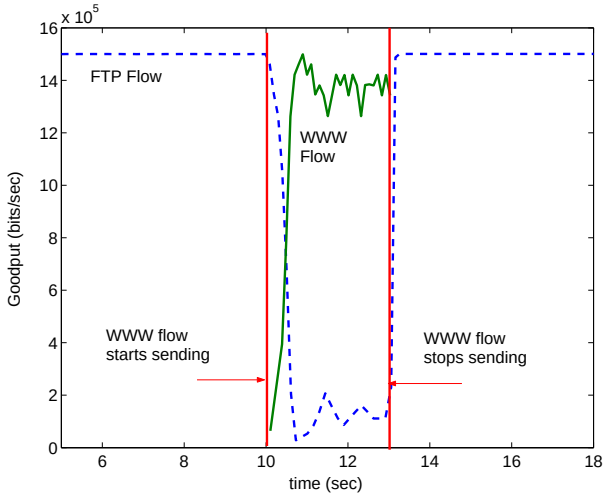

Figure 7: Rates allocated to two application streams. Background file transfer is interrupted by a WWW session that starts at time 10 seconds, when it obtains the majority of the bandwidth, until it terminates at time 13 seconds.

with greater control over how applications' network connections share the available wireless capacity.

To design an efficient wireless network it is insufficient to focus on lower layer optimizations. The interaction of mechanisms operating at each layer must be taken into consideration. Heterogeneous network design is inherently a multi-layer discipline, as this article has demonstrated.

References

[1] P. Viswanath, D.N.C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1277–1294, June 2002.

[2] D. Tse and S. Hanly, "Multiaccess fading channels - Part I: polymatroid structure, optimal resource allocation and throughput capacities," *IEEE Trans. Inform. Theory*, vol. 4, no. 7, November 1998.

[3] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and S. Viterbi, "CDMA/HDR: a bandwidth efficient high speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, July 2000.

[4] X. Liu, E.K.P. Chong, and N.B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Select. Areas Commun.*, vol. 19, no. 10, pp. 2053 –2064, October 2001.

[5] S. Nanda, K. Balachandran, and S. Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, no. 1, pp. 54–64, January 2000.

[6] Dah-Ming Chiu and Raj Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[7] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, 1997.

[8] Lampros Kalampoukas, Anujan Varma, and K. K. Ramakrishnan, "Explicit window adaption: A method to enahance TCP performance," *IEEE/ACM Trans. Networking*, vol. 10, no. 3, pp. 338–350, June 2002.

[9] Lachlan L. H. Andrew, Stephen V. Hanly, and Rami G. Mukhtar, "CLAMP: Differentiated capacity allocation in access networks," in *Proc. IEEE Int. Performance Computing and Communications Conf.*, Phoenix, April 2003, pp. 451–458.