A QoS-Aware AIMD Protocol for Time-Sensitive Applications in Wired/Wireless Networks

Lin Cai, Xuemin (Sherman) Shen, Jon W. Mark
Centre for Wireless Communications
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada
Email: {*cai,xshen,jwmark*}@bbcr.uwaterloo.ca

Jianping Pan
NTT MCL
Palo Alto, CA, USA
Email: jpan@nttmcl.com

***Abstract*— A TCP-friendly Additive Increase and Multiplicative Decrease (AIMD) protocol is proposed to support time-sensitive applications in hybrid wired/wireless networks. By analyzing the performance of AIMD-controlled flows in hybrid networks, we propose a cross-layer procedure to select the AIMD protocol parameters with consideration of wireless link characteristics and application QoS requirements, in terms of delay, loss, and throughput. Since the cross-layer interaction only exchanges parameters among the application, the transport layer protocol, and the link layer protocol, our approach preserves the end-to-end semantics of the transport protocol and the layered structure of the Internet, and it is applicable to supporting various multimedia applications over heterogeneous wireless links. With appropriate parameters, AIMD-controlled flows can fairly share network resources with TCP flows, efficiently utilize wireless resources, and statistically guarantee end-to-end delay for time-sensitive applications. Extensive simulations are performed to validate the analytical results, evaluate the protocol performance, and demonstrate that the AIMD protocol can outperform the unresponsive UDP protocol when transporting multimedia traffic in hybrid networks. With satisfactory QoS, end systems have more incentives to voluntarily regulate multimedia traffic with an AIMD-based congestion controller, which is vital for network stability, integrity, and future proliferation.**

I. Introduction

The Internet and wireless cellular networks will converge to a ubiquitous all-IP information infrastructure, allowing users to access the hybrid wired/wireless networks for multimedia services anywhere, anytime. Traditionally, the IP-based Internet and wireless cellular networks have different resource management approaches. In the Internet, under the control of the transport protocol, coexisting flows share network resources in a distributed manner. In wireless networks, to maintain the QoS of existing and handoff calls with limited wireless bandwidth, centralized resource management and allocation schemes are used. Therefore, in hybrid networks, end-to-end transport protocols should regulate the cross-domain traffic to efficiently and fairly share highly multiplexed wired links in a distributed manner, and to maximize the resource utilization of lowly multiplexed or dedicated wireless links.

On the other hand, hybrid networks are anticipated to provide multimedia services, so transport protocols should also provide a wide variety of QoS to heterogeneous multimedia applications. However, TCP, the currently dominant transport protocol, has been designed and engineered mainly for bulk data transfer. In this paper, the acronym TCP refers to TCP SACK [1]. TCP uses a congestion window (*cwnd*) to probe for available bandwidth and respond to network congestion, according to the AIMD congestion control mechanism: TCP sender increases its *cwnd* by one segment per round-trip time (*rtt*) if no congestion signal is captured, and decreases the *cwnd* by half otherwise. With TCP's increase-by-one or decrease-by-half control strategy, even an adaptive and scalable source coding scheme cannot hide the flow throughput variation; therefore, the user-perceived quality may change ungracefully. In addition, TCP offers a reliable data transfer service, which may introduce intolerable delay and delay jitter for time-sensitive multimedia applications. For these reasons, TCP is not suitable for multimedia applications.

Therefore, TCP-friendly congestion control for multimedia applications has become an active research topic [2]–[8]. Two paradigms of TCP-friendly congestion control mechanisms have been proposed: equation-based rate control, *e.g.*, TCP-Friendly Rate Control (TFRC) [4], and window-based binomial control, *e.g.*, AIMD [6]–[8]. On the other hand, for applications not requiring reliable service, Datagram Congestion Control Protocol (DCCP) has been proposed, which combines unreliable datagram delivery with built-in congestion control [9]. DCCP includes different TCP-friendly congestion control algorithms: TCP-like congestion control, TFRC, *etc.*

However, how to efficiently control the cross-domain multimedia traffic with *end-to-end* QoS provisioning is still an open issue. End-to-end delay guarantee for time-sensitive applications over wireless networks is especially challenging, due to the time-varying and error-prone wireless channel characteristics. This paper is devoted to meeting this challenge.

We focus on the AIMD-based DCCP protocol (named AIMD protocol), which is extended from the TCP-like DCCP protocol (DCCP-2) [10]. The AIMD protocol inherits the same AIMD congestion control mechanism as that of TCP. Instead of the "increase-by-one or decrease-by-half" strategy, the AIMD sender increases its *cwnd* by α segments additively when no congestion is sensed; otherwise, it multiplicatively decreases the *cwnd* to β times its previous value [6]–[8]. The (α, β) pair can be flexibly chosen by applications, under the constraint of the TCP-friendly condition derived in [6], [8]. With the same control mechanism as that of TCP, the AIMD

0-7803-8968-9/05/$20.00 (C)2005 IEEE

protocol is compatible with the legacy and scalable to be deployed incrementally. In addition, as discussed in a later section, window-based AIMD protocol has the *acknowledgment self-clocking* property, which is particularly useful to regulate traffic over time-varying wireless links.

The main contributions of this paper are as follows. The TCP-friendly AIMD protocol is proposed to support multimedia applications. Based on the QoS performance analysis of AIMD-controlled flows in hybrid networks, we demonstrate how to select the protocol parameters with consideration of the wireless link characteristics and application QoS requirements. Since the cross-layer design requires only the exchange of parameters among the application, the transport layer protocol, and the link layer protocol, our approach preserves the end-to-end semantics of the transport protocol and the layered structure of the Internet, and it is applicable to supporting various multimedia applications over heterogeneous wireless links. Finally, extensive simulations with NS-2 [11] are performed to validate the analysis, demonstrate the feasibility of our approach, and show that the TCP-friendly AIMD protocol can outperform the unresponsive UDP protocol for time-sensitive applications.

The remainder of the paper is organized as follows. Section II introduces the AIMD protocol. Based on the QoS performance analysis in Section III, we demonstrate how to appropriately select the protocol parameters in Section IV. In Section V, simulation results are given to validate the analytical results, evaluate the protocol performance, and compare the QoS performance of the AIMD protocol with that of the UDP protocol. Related work is discussed in Section VI, followed by concluding remarks in Section VII.

II. AIMD Protocol

Extended from DCCP-2, the AIMD protocol uses a window-based flow and congestion control mechanism. The AIMD sender sends packets[1] whose sequence numbers are in the range of a sliding window, the sender window; the AIMD receiver sends acknowledgments (*acks*) for correctly received packets. The sender window size, W, is determined and dynamically adjusted by the flow and congestion control mechanisms.

In the following, we first introduce the main protocol components: the acknowledgment scheme and the flow control and congestion control mechanisms. Then, we discuss why the window-based TCP-friendly AIMD congestion control mechanism is chosen. Due to limited space, other protocol details are referred to the specification of DCCP-2 [10].

A. AIMD Acknowledgment

Since the AIMD sender is not obligated to retransmit corrupt or lost packets, the cumulative acknowledgment scheme used in TCP is not applicable. We use a simple acknowledgment scheme for the unreliable AIMD protocol. The AIMD *ack* has two fields: a 24-bit acknowledgment number, ack_a, and a selective acknowledgment vector with negotiable length, $sackvec$. ack_a identifies the packet with a valid sequence number received from the sender; the i-th bit in $sackvec$ indicates the status (whether or not it has been received) of the packet with sequence number $(ack_a - i)$. The length of $sackvec$ is denoted as $|sackvec|$.

With $sackvec$, the AIMD *acks* have some redundancy and can tolerate occasional losses of *acks*. The AIMD receiver can send one *ack* for several (up to $1+|sackvec|$) packets received if the bandwidth consumption of *acks* is of great concern. There is no guarantee that the AIMD sender know all packets' status from *acks*. For a packet not being indicated in any *ack* within certain time, it is assumed lost (which is a timeout event). Thus, our scheme does not require the AIMD sender to acknowledge *acks*, and the receiver does not retransmit *acks*.

B. Flow and Congestion Control

To avoid a fast sender overrunning a slow receiver, the AIMD receiver advertises the amount of the allocated buffer for the connection, and the AIMD sender uses the receiver's advertised window (*rwnd*) to bound the amount of in-flight packets.

The AIMD receiver maintains a buffer of size *rwnd*, and the buffered packets are always the newest ones. Those received packets with sequence number less than the "largest" ("largest" is measured in circular sequence space) sequence number minus *rwnd* are discarded without any *ack*; otherwise, the received packets are buffered, and the receiver sends *acks* for them. Since each *ack* indicates the status of $|sackvec| + 1$ packets, the receiver needs to store $rwnd + |sackvec| + 1$ packets' status information.

The congestion control mechanism used in the AIMD protocol is similar to that in TCP SACK [1], except that a pair of parameters, (α, β), is introduced. Specifically, to probe for available bandwidth and respond to network congestion, the AIMD sender uses a *cwnd* to control the sending rate. The actual size of sender window (W) is the minimum of *cwnd* and *rwnd*.

The *cwnd* evolves in three stages to converge to the optimal operation region. Initially, after a timeout, or being idle for a while, the *cwnd* is set to a small initial window (*IW*), and it is doubled each *rtt*, which is the *slow start* stage. The slow start threshold (*ssthresh*) is set to reflect the estimated available bandwidth. When *cwnd* exceeds *ssthresh*, *cwnd* is additively increased by α packet per *rtt*, which is the *congestion avoidance* stage, until eventually congestion occurs. Severe congestion is indicated by timeout, which forces the AIMD sender to reinitialize the *cwnd* and halve the *ssthresh*, followed by slow start. Moderate congestion is indicated when the AIMD *acks* show that one packet is not received and three or more packets with larger sequence numbers are received, and the former packet is assumed to be lost. When this occurs, the *cwnd* is reduced by a factor of β (or more precisely, reduced

[1] Modern transport protocols can negotiate the maximum segment size on its connection establishment to avoid IP fragmentation. Link layer fragmentation is not considered here, since time-sensitive multimedia traffic usually has small packet size. In the sequel, the term *packet* is used to represent link *frame*, network *packet*, and transport *segment*.

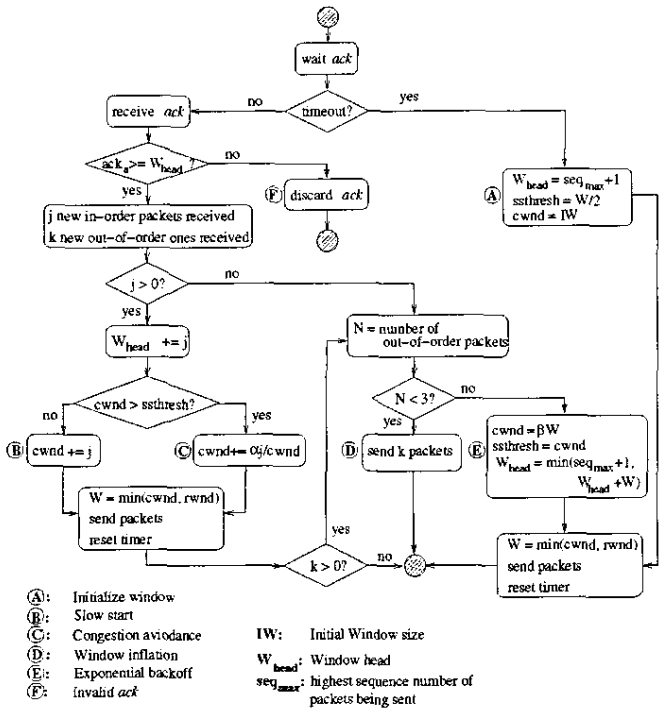


Fig. 1. AIMD sender

to β times the number of currently outstanding packets) and $ssthresh = cwnd$, which is the *exponential backoff* stage. Similar to TCP, the AIMD sender will backoff only once for all packet losses within one window.

The flow and congestion control algorithms at the AIMD sender side are shown in Fig. 1. If timeout occurs, the AIMD sender re-initializes the *cwnd* and W, advances the head of the sender window (W_{head}), and sends W new packets (event **A** in Fig. 1); if the current *cwnd* is one, the new *cwnd* is still one, but the inter-packet spacing is doubled such that the sending rate is halved. If an *ack* is received and it indicates that j packets with sequence number $W_{head}, \cdots, W_{head}+j-1$ are received successfully, the *cwnd* is enlarged by j if in slow start stage (event **B**), or enlarged by $\alpha j/cwnd$ if in congestion avoidance stage (event **C**). If the *ack* indicates that k out-of-order packets (with sequence numbers larger than W_{head}) are received, the window is inflated by k and k new packets are sent if the number of out-of-order packets is less than three (event **D**); otherwise, the window is deflated, the *cwnd* is decreased by a ratio of β, and W_{head} is advanced to guarantee that the window is reduced only once per *rtt* (event **E**). If ack_a is less than W_{head}, the sender discards the *ack* (event **F**). To avoid a burst of packets being pumped into the network, the inter-packet spacing is set to be no less than rtt/W when the sender is allowed to send more than one packet at certain time[2].

[2]The packet spacing technology has been deployed in Rate Adaptation Protocol (RAP) [3] and TCP rate control [12].

C. Advantages of Window-based AIMD Mechanism

The window-based AIMD congestion control mechanism is chosen, because: a) the success of the Internet over the past two decades has demonstrated the effectiveness and efficiency of the AIMD mechanism; b) it has been shown in [13] that, in general, anything slower than exponential backoff cannot guarantee network stability when the end systems have no complete knowledge of the global traffic; c) Loguinov and Radha [14] have shown that AIMD is the *only* TCP-friendly binomial control with monotonic convergence to fairness, and under the TCP-friendly condition derived in [8], AIMD and TCP flows can fairly share link bandwidth, regardless of the link capacity and the number of coexisting flows; and d) the increase rate and decrease ratio (α, β) pair can be flexibly chosen to provide a wide variety of QoS to various multimedia applications, *e.g.*, for applications preferring smooth throughput, a large value of β and a small value of α can be chosen.

In addition, window-based protocols have the *acknowledgment self-clocking* property, which is particularly useful to regulate traffic over time-varying wireless links. In wireless links, the transmission error rate is non-negligible. For a given wireless channel and Forward Error Correction (FEC) coding scheme, the residual transmission errors are still visible to upper layer protocols, and the instantaneous error rate fluctuates from time to time. To *hide* transmission errors from the upper layers, an effective error recovery scheme widely deployed in wireless links is the Automatic Repeat reQuest (ARQ) scheme [15]: the link layer detects transmission errors and performs retransmissions locally. With the link level ARQ, the wireless link throughput is time-varying. When the wireless channel is in poor condition temporarily, the AIMD sender can immediately reduce the sending rate since the *ack*s are delayed due to low link throughput. Once the channel condition becomes better, the *ack*s can arrive at the sender at a faster pace, and the sending rate will be increased accordingly. Therefore, queue length at the wireless link is always constrained by window size.

III. Performance Analysis

To support time-sensitive multimedia applications, the performance of AIMD-controlled flows in hybrid networks should be understood.

A. QoS Performance Indexes

For time-sensitive multimedia applications over best-effort and highly dynamic IP-based networks, error-resilient and rate-scalable source coding schemes have been proposed and are anticipated to be widely deployed [2], [16]–[18]. For instance, in the Amendment of MPEG-4, fine granularity scalability (FGS) video-coding technique is developed for streaming video over the Internet [16], [17]. With the FGS technique, a video sequence is coded into a non-scalable base layer and a scalable enhancement layer. A certain degree of packet losses in the enhancement layer is tolerable; but packet losses in the base layer may result in severe distortion of the video, and they are intolerable. Another approach is

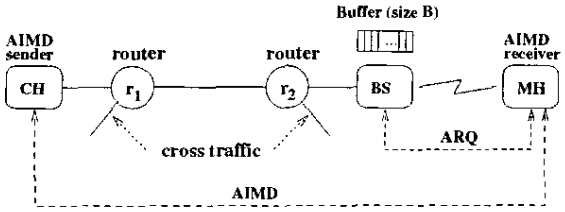


Fig. 2. System model

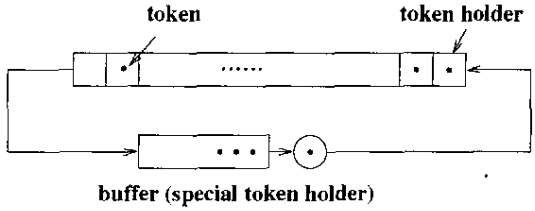


Fig. 3. Token emulation, single flow

the non-layered multiple description (MD) coding [18]. An MD coder partitions the source data into several sets and then compresses independently to produce descriptions. The quality of reconstructed multimedia can be improved when more descriptions are received, and the MD decoder can conceal packet losses up to a certain degree. Since the Internet currently cannot provide end-to-end differentiated services, the non-layered MD coding is more attractive for Internet-based multimedia applications, and it is adopted in our system.

With the MD technique, given the sending rate in the transport layer, the source encoder can determine an optimal bit-rate and the maximum tolerable packet loss rate. Application-perceived packet loss rate measures the ratio of packets failed to arrive at the multimedia receiver *timely*. For time-sensitive applications, packets suffering excessive delay become useless, and they will be discarded by the receiver, which is called delay outage events. Besides packet losses in the network, delay outage rate, the ratio of received packets with delay jitter (mainly due to queuing delay d_q) larger than a prescribed threshold D, is an important component of application-perceived packet loss for time-sensitive applications. In the sequel, packet loss rate is defined as the ratio of packets lost in the network, and delay outage rate is the ratio of packets received with excessive delay.

In summary, the QoS parameters considered are flow throughput, packet loss rate, and delay outage rate. Transport layer protocols should regulate traffic appropriately to maximize the flow throughput and bound the packet loss rate and the delay outage rate.

B. System Model

We consider the scenario shown in Fig. 2. Let a multimedia connection be established between a correspondent host (CH) and a mobile host (MH), through a last hop wireless link between a base station (BS) and the MH. The cross-domain connection is controlled by the AIMD(α, β) protocol. Let the AIMD receiver acknowledge each received packet without delay, and the AIMD sender be saturated. Assume that all packets of the target flow have the same packet size and travel through the same route.

In the wired domain, the target flow shares the link with other cross traffic. In the wireless link with link level ARQ, corrupted packets will be retransmitted *immediately* after the current transmission, (the instantaneous-retransmission assumption can be eliminated, and the detail has been given in the extended technical report [19]). For analysis simplicity, assume that *ack*s are transmitted without error (due to the redundancy in AIMD *ack*s, the flow performance will not be affected significantly with occasional losses of *ack*s). Assume that the AIMD sender sets end-to-end round-trip timers conservative enough to allow sufficient time for the link level ARQ to recover from transmission errors. This assumption tends to hold, since the timeout value is estimated as the average of measured *rtt*s plus a conservative factor proportional to the measured standard deviation. Furthermore, coarse-grained timers with a granularity of 500 ms are implemented in practice. With the link level ARQ, the wireless link throughput, defined as the number of packets being successfully transmitted per unit time, is a time-varying random variable. The link throughput distribution can be calculated given the wireless channel profile and the link layer packet transmission scheme [19], which is beyond the scope of this paper.

In our analytical framework, time is discretized into slots. Each slot is long enough to transmit one packet over the wireless link. Let R slots denote the *rtt* without the queuing delay and retransmissions over the wireless link. R can be approximately constant, since the wireless link is presumably the bottleneck, and queuing delay over the wireless link is dominant and can absorb delay jitter in the wired domain.

In the following, we first consider the scenario that one AIMD flow occupies a dedicated wireless link. Then, we extend our study to the multiple AIMD flows scenario. We focus on the flow throughput, packet loss rate due to BS buffer overflow, and delay outage probability. For delay analysis, since end-to-end delay is mainly related to the window size during the past *rtt*, we need to obtain the delay performance given a window size.

C. Single AIMD Flow

We use the concept of *token* to emulate the window-based AIMD control, as shown in Fig. 3. A packet, which can be data or *ack*, needs a token to traverse the network. The sender has W tokens in total. When an *ack* returns to the sender with a token, a new data packet can be sent with the token. Once the receiver receives a new packet, it uses the token to send out an *ack*. Only the sender can adjust the number of tokens according to the flow and congestion control mechanism. It takes R slots for a token finishing the round trip if there is no queuing delay and retransmission at the BS. The round-trip path is discretized into R token holders. A token shifts to the next holder along the path per time slot (similar to shift register), unless it is in the BS buffer, which is a special token

holder. If a packet is transmitted successfully over the wireless link, the token with the packet can shift from the BS buffer to the next token holder.

The BS buffer size is denoted as B. We consider two cases: $B \geq W$ and $B < W$.

Case 1: $B \geq W$, no buffer overflow occurs at the BS. For a flow with window size W, if during the past $R-1$ slots, n packets were transmitted successfully over the wireless link (*i.e.*, n tokens outside the BS buffer), the BS queue length (the number of tokens in the buffer), Q, equals $(W-n)^+$, where $0 \leq n \leq R-1$ and $(x)^+ = \max(x, 0)$.

To maximize the link utilization and flow throughput, the wireless link should not be idle whenever the link layer decides to transmit, so the BS queue should be non-empty all the time. Therefore, the sufficient condition to maximize the link utilization and flow throughput is $W \geq R$. The maximum flow throughput is $1-p_e$ packet per slot, where p_e is the average packet error rate over the wireless link.

When $W \geq R$ and $B \geq W$, the delay outage probability equals the probability of less than W packets being successfully transmitted over the wireless link in $R+D$ slots:

$$\Pr\{d_q > D|W\} = \sum_{x=0}^{W-1} T_{(R+D)}(x), \tag{1}$$

where $T_n(x)$, the wireless link throughput distribution, is the probability of having x successful transmissions over the wireless link in n slots.

Case 2: $B < W$, buffer overflow may occur. When $B < W$ and $W \geq R$, the probability of packet loss due to BS buffer overflow, $L(B, W)$, equals the probability that less than $W-B$ successful transmissions over the wireless link in the past $R-1$ slots: $L(B,W) = \sum_{x=0}^{W-B-1} T_{R-1}(x)$.

The queue length distribution is $\Pr\{Q = x|W\} = T_{R-1}(W-x)$, where $(W-R+1) \leq x \leq B$. When a packet arrives at the queue with length x, the probability of its queuing delay exceeding D slots is $\sum_{i=0}^{x-1} T_{D+1}(i)$. Thus, the delay outage probability in the second case is given by:

$$\Pr\{d_q > D|W;B\} = \sum_{x=W-R+1}^{B} \sum_{i=0}^{x-1} T_{D+1}(i) T_{R-1}(W-x). \tag{2}$$

D. Multiple AIMD Flows

Let N AIMD flows share a wireless link. For the i-th flow, the *rtt* without queuing delay and retransmissions over the wireless link is denoted as R_i, and its sender window size is W_i. Without loss of generality, let $R_1 \leq R_2 \leq \cdots \leq R_N$. Denote p_i the probability that the packet being transmitted over the wireless link is from the i-th flow. The mean number of the i-th flow's tokens in the BS buffer equals $W_i - \mathrm{E}[T_{R_i-1}]p_i$, where $\mathrm{E}[T_{R_i-1}]$ is the mean number of successful transmissions over the wireless link in R_i-1 slots. On the other hand, p_i equals the ratio of the i-th flow's tokens in the buffer:

$$p_i = \frac{W_i - \mathrm{E}[T_{R_i-1}]p_i}{S - \sum_{j=1}^{N} \mathrm{E}[T_{R_j-1}]p_j}, \tag{3}$$

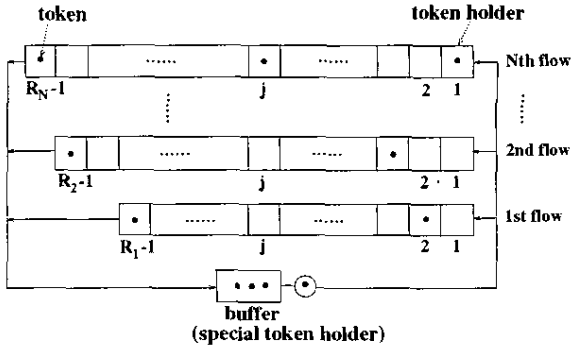


Fig. 4. Token emulation, multiple flows

where $S = \sum_{i=1}^{N} W_i$.

From (3), $p_i/p_j = (W_i - \mathrm{E}[T_{R_i-1}]p_i)/(W_j - \mathrm{E}[T_{R_j-1}]p_j)$. Therefore, to assign the i-th flow a p_i portion of the wireless bandwidth, W_i should satisfy the following condition:

$$W_i = p_i W_j/p_j + (\mathrm{E}[T_{R_i-1}] - \mathrm{E}[T_{R_j-1}])p_i, \tag{4}$$

for $1 \leq i, j \leq N$.

Equation (4) gives the *necessary and sufficient* condition of proportional fairness for window-controlled flows. According to (4), for $p_i = 1/N$, flows' window sizes are not proportional to their *rtt*s. To achieve fairness for flows with different *rtt*s, the number of coexisting flows should be taken into account. This reveals why AIMD mechanism alone cannot achieve fairness for flows with different *rtt*s, no matter how (α, β) pair is chosen (such bias has been reported in the literature [20]). The simulation results in Section V-C validate (4).

As shown in Fig. 4, the path of each flow corresponds to a set of token holders. The j-th token holder of the i-th flow holds one token if an i-th flow's packet was successfully transmitted over the wireless link j slots ago. Since the sum of tokens in the j-th token holders of all N flows is either zero or one, the maximum number of in-flight tokens outside the BS buffer is $R_N - 1$, and the BS queue length is no less than $(S - R_N + 1)^+$. To fully utilize the wireless link, the sufficient condition is $S \geq R_N$, and the maximum throughput of the i-th flow is $(1-p_e)p_i$ packet per slot.

Denote $T_n(x; p_i)$ as the probability of successfully transmitting x packets of the i-th flow in n slots. $T_n(x; p_i)$ can be calculated as:

$$T_n(x; p_i) = \sum_{k=0}^{n-x} \binom{x+k}{x} p_i^x (1-p_i)^k T_n(x+k), \tag{5}$$

where $0 < p_i < 1$. Obviously, $T_n(x) = T_n(x; 1)$, and $T_n(x; p_i) + T_n(x; p_j) = T_n(x; p_i + p_j)$.

The number of i-th flow's tokens outside the buffer equals the number of i-th flow's packets being successfully transmitted in the past $R_i - 1$ slots. Thus, the queue length distribution is

$$\Pr\{Q = x|\boldsymbol{W}\} = \sum_{y_1+\cdots+y_N=S-x} \cdots \sum [T_{R_1-1}(y_1; 1) \cdot T_{R_2-R_1}(y_2; 1-p_1) \cdots T_{R_N-R_{N-1}}(y_N; p_N)], \tag{6}$$

where $\boldsymbol{W}$ represents the vector $\{W_1, W_2, \cdots, W_N\}$, and $S - R_N + 1 \leq x \leq S$.

Consequently, when $B \geq S$ (there is no buffer overflow), the delay outage probability is given by

$$\Pr\{d_q > D|\boldsymbol{W}\} = \sum_{x=S-R_N+1}^{S} \Pr\{Q = x|\boldsymbol{W}\} \sum_{j=0}^{x-1} T_{D+1}(j), \tag{7}$$

where $S \geq R_N$ and $\Pr\{Q = x|\boldsymbol{W}\}$ is given in (6).

When $B < S$, the probability of packet loss due to buffer overflow is

$$L(B, \boldsymbol{W}) = \sum_{x=B+1}^{S} \Pr\{Q = x|\boldsymbol{W}\}, \tag{8}$$

and the delay outage probability is

$$\Pr\{d_q > D|\boldsymbol{W}; B\} = \sum_{x=S-R_N+1}^{B} \Pr\{Q = x|\boldsymbol{W}\} \sum_{j=0}^{x-1} T_{D+1}(j), \tag{9}$$

where $S \geq R_N$ and $\Pr\{Q = x|\boldsymbol{W}\}$ is given in (6).

In summary, the delay outage probability and the BS buffer overflow probability are a function of the AIMD sender window size and the wireless link throughput distribution. The latter can be calculated given the wireless channel profile and the link layer transmission scheme [19]. Since the QoS analysis is isolated from the link throughput, our analytical framework can be extended to consider other link layer protocol features, *e.g.*, non-persistent link level ARQ scheme, by incorporating them in the derivation of the link throughput distribution.

IV. Parameter Selection

To efficiently support multimedia applications in hybrid networks, the design objective is to be TCP-friendly in wired links, maximize the wireless link utilization and flow throughput, and bound the packet loss rate and delay outage probability.

A. TCP-friendliness

TCP-friendliness is defined as the average throughput of non-TCP-transported flows over a large time scale does not exceed that of any conformant TCP-transported ones under the same circumstances [21]. It has been shown in [8] that AIMD parameters satisfying the following condition can guarantee TCP-friendliness, no matter what the bottleneck link capacity is and how many TCP and AIMD flows coexist in the link:

$$\alpha = 3(1 - \beta)/(1 + \beta), \tag{10}$$

where $0 < \alpha < 3$ and $0 < \beta < 1$. Different applications can choose one of the parameters, and the other parameter is determined by (10). (TCP-friendly AIMD still tends to bias against long *rtt* flows, as TCP does, unless the coexisting flows' windows satisfy (4).)

B. QoS Provisioning for Single AIMD Flow Case

AIMD flows probe for available bandwidth and overshoot the bottleneck link capacity frequently, which produces transient congestion and packet losses. For highly multiplexed bottleneck, dynamic probing with AIMD mechanism is required since end systems do not have the knowledge of global traffic. However, for a cross-domain connection, the bottleneck is most likely the lowly multiplexed wireless link, and, in general, dedicated wireless links are allocated to multimedia flows. The dynamics of available bandwidth in wireless links may not be due to the competition of multiplexed flows, but due to the time-varying wireless channel condition. Overshooting a dedicated wireless link frequently is not an efficient way to utilize it.

As shown in Section III-C, the sufficient condition to maximize the wireless link utilization and flow throughput is $W \geq R$, where $W = \min(rwnd, cwnd)$. Obviously, $rwnd \geq W$. Less obviously, we can choose appropriate $rwnd$ and B such that W converges to $rwnd$: $W = rwnd \geq R$.

When $B < W$, if the number of successful transmissions in $R-1$ slots is less than $W - B$, buffer overflow at the BS may occur. Packet losses due to poor channel condition can trigger the AIMD sender to exponentially reduce W. Consequently, when channel condition becomes better later, there may not be enough packets for transmission due to the small window size. Thus, the BS buffer size, B, can be conservatively set to $rwnd$ to avoid buffer overflow due to wireless channel dynamics[3]; the exponential backoff will not be triggered unnecessarily.

Ideally, if there is no delay jitter and packet losses in the wired domain, by setting $B = rwnd = R$, the AIMD sender window size converges to $rwnd$ in steady state, *i.e.*, $W = rwnd = R$. Thus, the wireless link is fully utilized and the flow throughput is maximized. The minimum queue length at the BS is one. There is no packet loss over the wireless link, and the delay outage rate is bounded according to (1).

In reality, there may be delay jitter and packet losses in the wired domain. With $B = rwnd$, there is still no packet loss over the wireless link, so the overall packet loss rate equals the packet loss rate in the wired domain, which can be estimated and bounded according to the historical measurements (*e.g.*, referring to the data in [23]). On the other hand, setting *rwnd* to R cannot absorb delay jitter in the wired domain, so that the wireless channel may be idle sometimes. For instance, when $rwnd = R$, if one packet is delayed one more slot in the wired domain, the BS queue may be empty for one slot. Such underutilization of wireless links can be avoided if $rwnd > R$. Also, when a single packet loss occurs in the wired networks, the AIMD sender will exponentially reduce its sender window. If β times *rwnd* is less than R, the queue at the BS will be empty for some slots, and the wireless link is underutilized. Therefore, to efficiently utilize the wireless link, it is better to set a larger *rwnd*.

However, a larger *rwnd* leads to a larger queue length and

[3]This is achievable, *e.g.*, Enhanced GRPS system can set the buffer size up to 48KB [22].

higher delay outage probability. The delay outage probability, given in (1), is a non-decrease function of the window size. The maximal window size satisfying the maximum tolerable delay outage probability, $W_{\max}$, is the optimal value of *rwnd*, which can maximize the wireless link utilization under the constraint of the delay outage bound.

C. QoS Provisioning for Multiple AIMD Flows Case

For multiple AIMD flows sharing the wireless link, to assign a ratio of wireless bandwidth to AIMD flows with the same or different *rtt*s, their window sizes should satisfy (4). The BS buffer size can be set to S to avoid buffer overflow, so the packet loss rate equals the estimated packet loss rate in the wired domain. The delay outage probability is given in (7). Under the constraint of (4), optimal *rwnd*s can be set to the maximum integers satisfying delay outage bound.

D. Parameter Selection Procedure

The parameter selection procedure is given as follows.

1) The application identifies its required throughput, maximum tolerable delay jitter D, delay outage probability, and packet loss rate; and it also selects a desired β.
2) The application passes these parameters to the transport layer protocol.
3) In the transport layer, given β, the AIMD sender calculates α according to the TCP-friendly condition (10), and measures and estimates the minimum *rtt* of the flow.
4) The AIMD sender sends the minimum *rtt* and the QoS parameters to the BS.
5) According to the wireless link characteristics and the physical layer protocol, the BS resource allocation module (in the link layer) calculates the packet-level wireless channel profile, determines the optimal link layer transmission scheme, and allocates wireless channels to the connection such that the average link throughput is no less than the desired flow throughput.
6) The BS calculates $W_{\max}$, the optimal *rwnd*, according to the QoS requirements and wireless link throughput distribution, and set $B = W_{\max}$.
7) The BS informs the AIMD receiver to set the allocated buffer size to $W_{\max}$. Then, the AIMD receiver informs the sender (by *ack*) that the *rwnd* is $W_{\max}$.

The computations involved in steps 3 and 5 are very light. The computations in step 6 can be done offline and the results can be stored in a look-up table, so the BS can check the table to determine appropriate *rwnd* and B quickly. Therefore, the parameter selection procedure can be done efficiently during the connection establishment phase. Steps 5 to 7 will be repeated when handoff occurs, or when the wireless channel profile changes significantly (as shown in Section V-A, optimal *rwnd* is insensitive to small changes of the channel profile).

To efficiently support multimedia applications with heterogeneous QoS requirements in hybrid wired/wireless networks, inter-layer interactions are necessary. Nevertheless, such interactions should be minimized for scalable system design purpose. As shown in the above procedure, the cross-layer procedure requires only the (infrequent) exchange of QoS and protocol parameters among the application, the transport layer protocol, and the link layer protocol. Therefore, our approach preserves the end-to-end semantics of the transport protocol and the layered-structure of the Internet, and it is applicable to supporting various multimedia applications with a wide variety of QoS requirements over heterogeneous wireless links with different channel models and physical/link layer protocols.

V. Performance Evaluation by Simulation

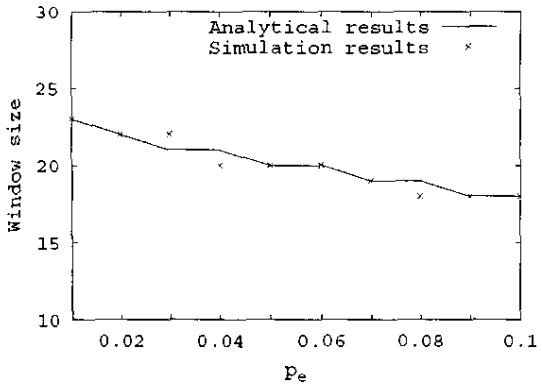
To verify the feasibility of our approach, examine link utilization, and evaluate the performance of the AIMD protocol, we have performed extensive simulations with NS-2 [11].

For one AIMD flow in the wireless link case, the simulation topology is shown in Fig. 2. For the target AIMD flow, the sender is at the CH, and the receiver is at the MH. The MH and the BS are connected to routers r_1 and r_2, respectively. Cross traffic (TCP SACK) connections share the r_1r_2 backbone link. The following parameters are used in the simulations unless otherwise explicitly stated. Links between CH, r_1, r_2, and BS are duplex links with 100 Mbps. Both r_1 and r_2 are Random Early Detection (RED) capable. The downlink and uplink bandwidth between the BS and the MH are 200 Kbps and 100 Kbps, respectively. The BS buffer size is set to *rwnd* to avoid buffer overflow. The minimal *rtt* for the target flow is 85 ms. The downlink channel condition is dynamically changed according to the finite-state Markov model. The TCP-friendly AIMD protocol has $\alpha = 0.2$ and $\beta = 0.875$. The target flow has packet size 125 bytes. Thus, the duration of a time slot is 5 ms. Each simulation lasts for 80 seconds, and different initial randomization seeds are used to reduce simulation dynamics. To eliminate system warming-up effects, simulation results for the first 5 seconds are not counted.

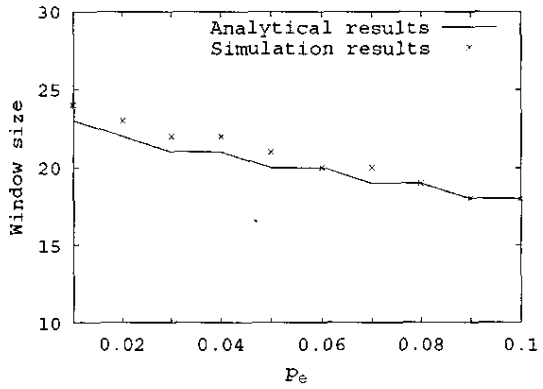
A. Delay Outage Rate

We repeat simulations with different values of *rwnd* and record the maximum one with which the delay outage rate is below 1%. Here, the delay outage rate is the ratio of packets with delay jitter exceeding 50 ms. To examine the effectiveness of the analysis in various scenarios, simulations with light cross traffic and heavy cross traffic have been performed. In the light cross traffic case, the cross TCP traffic (both long-lived elephants and short-lived mice) sharing the link r_1r_2 is constrained so that the bottleneck for the target AIMD flow is always the wireless link; in the heavy cross traffic case, the coexisting traffic volume is time-varying so that the bottleneck for the target flow is the backbone link between 30 s and 60 s. The link layer uses the persistent transmission scheme, *i.e.*, transmitting one packet each time slot no matter what the previous channel condition was.

Figs. 5(a) and (b) compare the analytical and simulation results of the maximum *rwnd*s with which the delay outage rate is less than 1%, for the light cross traffic and the heavy cross traffic cases, respectively. The wireless channel condition evolves following a two state Markov model, and the average packet error rate, (p_e), is set from 1% to 10%. It can be

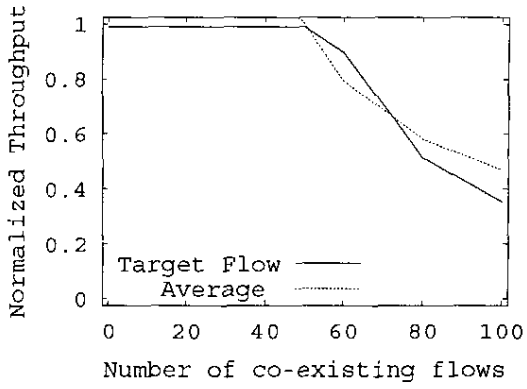


(a) light cross traffic

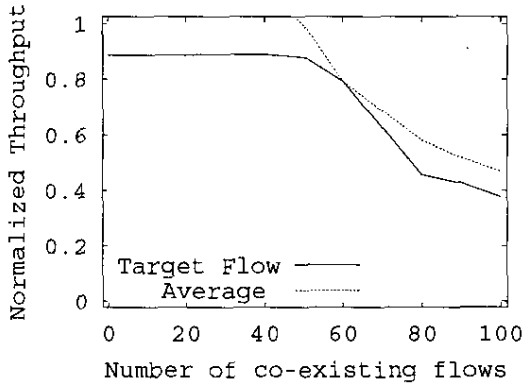


(b) heavy cross traffic

Fig. 5. Maximum *rwnd* to guarantee delay outage rate



(a) $p_e = 0.01$



(b) $p_e = 0.1$

Fig. 6. Normalized flow throughput

seen that the analytical results match well with the simulation ones with light cross traffic. As anticipated, the analytical results are slightly more conservative with heavy cross traffic since, when the bottleneck is the backbone link, *cwnd* and W are less than *rwnd*, and smaller window size corresponds to lower delay outage rate. Simulations with other link layer transmission schemes (not shown here due to page limitation) have been performed (*e.g.*, using the *optimal* transmission scheme which suspends transmitting for some slots if the previous transmission failed [24]), and the simulation results also match the analytical ones well. In summary, no matter whether or not the bottleneck link is the wireless link and what link layer transmission scheme is used, it is feasible to determine a suitable *rwnd* beforehand to bound the delay outage rate.

In addition, Fig. 5 shows that the optimal *rwnd* size changes slowly w.r.t. p_e. For instance, when p_e is changed from 5% to 8%, the optimal *rwnd* changes only one. Therefore, unless the channel profile changes significantly (*e.g.*, due to a sudden change of mobile pattern or wireless environment), the BS does not need to recalculate the optimal *rwnd* (repeating steps 5 to 7 of the parameter selection procedure), and our scheme can tolerate certain degree of errors in obtaining the wireless channel profile.

B. Link Utilization and TCP-friendliness

For congestion control in hybrid wired/wireless networks, ideally, the allocated wireless resources should be fully utilized if the bottleneck is the wireless link, and the cross-domain flow should occupy only its fair share of bandwidth when the bottleneck is the highly multiplexed backbone link, *i.e.*, being TCP-friendly.

To examine wireless link utilization and TCP-friendliness, the number of coexisting TCP flows (elephants) in the backbone link is changed from 0 to 100. The packet size of the cross TCP flows is 1250 bytes. Normalized flow throughput is defined as the number of packets being received successfully (in the transport layer) per time slot.

Figs. 6(a) and (b) plot the normalized throughputs of the target AIMD flow and the average normalized throughputs of

all coexisting TCP and AIMD flows, with p_e equal to 0.01 and 0.1, respectively. When the number of coexisting flows is less than 50, the average normalized throughput in the backbone link is larger than 1, and the bottleneck for the target AIMD flow is the wireless link. Therefore, the normalized throughput of the target flow should be $1-p_e$ to fully utilize the wireless link. When the number of coexisting flows is larger than 50, the bottleneck is the backbone link, and the normalized throughput of the target AIMD flow should be close to the average throughput in the backbone link in order to be TCP-friendly. It can be seen that the target AIMD flow demonstrates the desired performance. The simulation results with other p_e show the same tendency.

Meanwhile, the delay outage rates for all cases are below 1% as required. The packet loss rates are negligible when the number of coexisting flows is less than 50, and packet loss rates increase to 2.8% when the number of coexisting flows is increased to 100, which indicates that the packet losses are mainly due to network congestion in the wired domain. Therefore, with appropriate buffer size and window size, the packet loss over wireless domain and the delay outage rate can be bounded.

It is noticed that when the bottleneck link of the target flow is in the wired domain, the allocated wireless resources cannot be fully utilized. Our future work will study how the BS dynamically adjusts the allocated resources to a specific flow to improve the overall system performance.

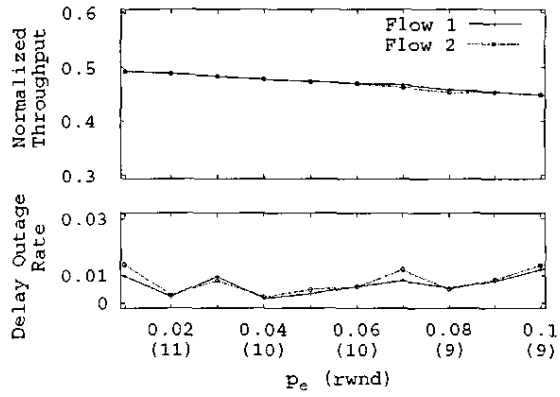
C. Multiple AIMD Flows

The simulation topology for multiple AIMD flows sharing a wireless link is similar to that shown in Fig. 2, except that N AIMD senders at N CHs connecting with r_1.

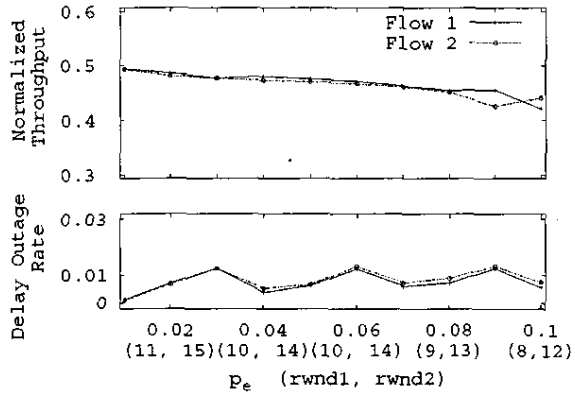
Let two AIMD flows have the same share of wireless resources. Their sender windows should satisfy (4). Also, to bound the delay outage rate, $\Pr\{d_q > D|W\}$ should be less than 0.01. The *rwnd*s are set as the maximum integers satisfying the above two constraints. The link buffer size is set to the sum of *rwnd*s to avoid buffer overflow at the BS. Fig. 7(a) shows the normalized throughputs and delay outage rates of two flows with the same minimal *rtt*s of 17 slots, w.r.t. p_e. In Fig. 7(b), two flows have different *rtt*s: $R_1 = 17$ and $R_2 = 25$. The simulation results demonstrate that the coexisting AIMD flows can fairly share the wireless link, and satisfy the delay outage bounds, no matter whether they have the same *rtt*s or not.

The same conclusion can be drawn when the number of flows sharing the wireless link is increased. Figs. 8(a) and (b) show the normalized throughputs and the delay outage rates for four coexisting AIMD flows. These four flows are designed to occupy the same share of wireless resources. In Fig. 8(a), all four flows have the same minimal *rtt*s of 17 slots; in Fig. 8(b), $R_1 = R_3 = 17$, and $R_2 = R_4 = 25$.

Since window size is an integer, we may not be able to get a group of *rwnd*s to satisfy (4) exactly. Therefore, the results may slightly deviate from our designed target due to quantization errors, *e.g.*, in some cases the throughputs of coexisting



(a) same *rtt*s



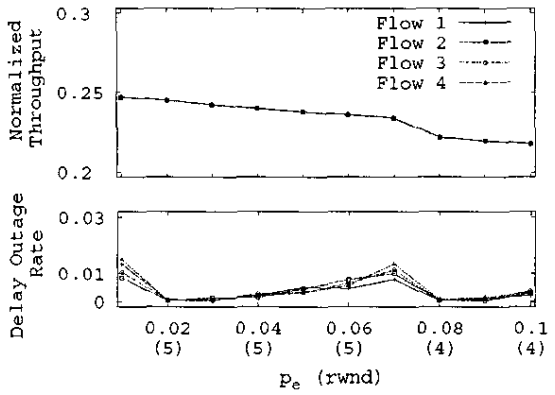
(b) different *rtt*s

Fig. 7. Two AIMD flows sharing the wireless link

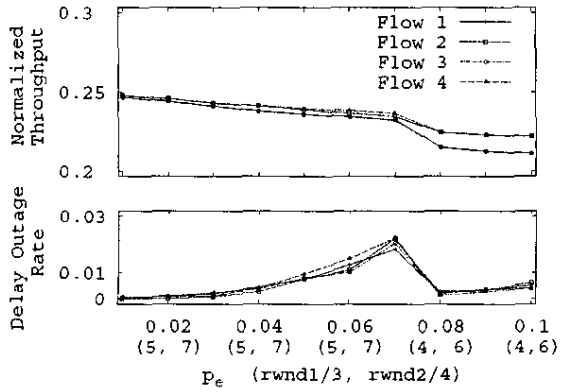
AIMD flows have a small difference, in some case the delay outage rates exceed the desired 1% slightly. Nevertheless, such deviations can be anticipated and controlled.

D. AIMD vs. UDP

The Internet has evolved from a small, research-oriented, and cooperative system to an enormous, commercial, and competitive information infrastructure. From selfish users' point of view, they would discard any congestion control in their systems if such control has negative effects on their perceived QoS. How to punish the greedy and malice is beyond the scope of this paper. However, by appropriately choosing the protocol parameters, the responsive AIMD protocol can outperform the unresponsive UDP protocol when supporting multimedia applications over wireless networks. This can be an incentive for end systems to deploy AIMD congestion control.



(a) same *rtts*



(b) different *rtts*

Fig. 8. Four AIMD flows sharing the wireless link

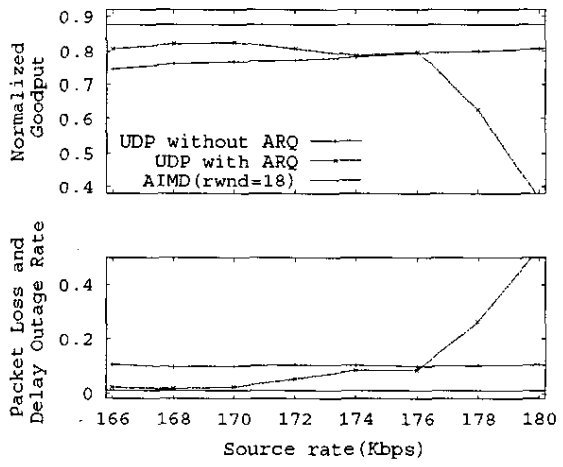


Fig. 9. AIMD vs. UDP. $p_e = 0.1$

Since the UDP protocol has no closed-loop control mechanism, the sender just keeps on sending at the source rate. Assume that the UDP sender has the knowledge of the wireless link, *i.e.*, link bandwidth, p_e, *etc.* , and it can determine an optimal sending rate accordingly. The BS can choose to use or not to use the link level ARQ for UDP traffic. Without ARQ, packet losses due to transmission errors is approximately p_e, which may be too severe when p_e is large. With ARQ, packet losses due to transmission errors can be avoided. However, unlike the AIMD sender which can slow down the transmission when the channel is in the poor condition due to its acknowledgment self-clocking property, the UDP sender cannot change the sending rate adaptively. Consequently, the BS queue is built up, and the queuing delay and delay outage probability increase quickly.

In the simulation, since the maximum tolerable delay jitter is 10 slots, to avoid excessive delay outage rate, the BS buffer size is set to 10 packets for UDP flows with constant bit rate (CBR) sources. The wireless link with and without ARQ are used for the UDP traffic, respectively. As a comparison, let an AIMD flow over the same wireless link with the link level ARQ, and the BS buffer size is set to *rwnd* for the AIMD flow. Normalized goodput is defined as the number of packets being successfully received with tolerable delay per slot, which is equivalent to the normalized flow throughput minus the delay outage rate.

Fig. 9 compares the normalized goodputs and packet loss and delay outage rates for UDP (with and without ARQ) and AIMD flows, with $p_e = 0.1$. Simulation results show that, without ARQ, the packet loss and delay outage rates for the UDP flow is approximately 0.1; this makes reconstruction of multimedia streams at the receiver more difficult. With ARQ, the source rate of the UDP flow should be less than the average wireless link throughput to avoid excessive delay outage rate. As shown in Fig. 9, no matter how the UDP sender adjusts its sending rate, and whether the wireless link deploys the ARQ for UDP traffic or not, the goodputs of the UDP flows are consistently lower than that of the AIMD flow, and the packet loss and delay outage rates of the UDP flows are consistently higher than that of the AIMD flow.

In summary, from the network service providers' point of view, the unresponsive UDP protocol may endanger network stability and integrity; from the users' point of view, the window-based AIMD protocol can provide better QoS (higher goodput and lower packet loss and delay outage rate).

VI. Related Work

Congestion control was incorporated into the TCP protocol in the late 1980's when a series of congestion collapses were observed even when the Internet was relatively small at that time [25]. The mainstream TCP variants, TCP Tahoe, TCP Reno, TCP New Reno, and TCP SACK, are all based on the AIMD congestion control mechanism, which uses packet losses as congestion signals with the assumption that packet losses are mainly due to network congestions. However, this assumption may not hold in the wireless domain which has

a noticeable transmission error rate. Efforts have been taken in both the link layer and the transport layer to improve TCP performance over wireless links.

Link layer approaches try to hide the transmission errors from TCP [26]–[28]. FEC coding is used to enhance the error correction ability by introducing more redundancy. However, wireless channels usually have burst errors, and FEC coding alone cannot efficiently correct burst errors. In general, for a given wireless channel and certain FEC coding scheme, the residual transmission errors of the decoding are taken care of by the link level ARQ. However, the link level ARQ introduces more delay variation, which should be taken into consideration when designing a transport protocol for time-sensitive applications. The work reported in this paper serves this purpose.

Besides the link layer approaches, many schemes have been proposed to adjust the TCP behavior over wireless links [29]–[33], either with or without the assistance of the interface node (the BS). With explicit congestion notification/explicit loss notification (ECN/ELN) [29], [30], the BS explicitly notifies the congestion or transmission losses to the TCP sender. Since ECN/ELN cannot recover transmission errors, either the link level ARQ or transport layer error recovery scheme is required when the application cannot tolerate excessive packet losses. For time-sensitive applications, the link level ARQ is preferable since it induces less delay and delay jitter. I-TCP is a split connection approach [32]: the BS establishes a standard TCP connection with the CH and a wireless TCP connection with the MH, so the controls are conducted and optimized over different domains. However, split connection approaches have significant overhead during handoff, and the end-to-end delay and delay jitter are difficult to control with two connections. With snoop TCP [31], the BS snoops the data packets and *acks*, and buffers the unacknowledged data. Once a packet loss due to transmission error is detected, the buffered packet is retransmitted locally. This approach is similar to the link level ARQ, except that the local retransmissions are performed in the transport layer. Since it may take longer time for the transport layer to detect transmission errors, for time-sensitive applications, it is preferable to use the link level ARQ to recover transmission errors. With M-TCP [33], the BS detects disconnections in the wireless domain and instructs the sender to freeze its timer and window. Once the connection regains, the sender is notified and can send at the full speed as that before disconnection. M-TCP is useful when temporary disconnection occurs due to deep fading channel condition or during the handoff period; M-TCP is orthogonal to our research and can be deployed in the proposed AIMD protocol to deal with temporary disconnections due to mobility. A survey on mobility support schemes is given in [34].

Since TCP is not suitable for emerging multimedia applications, two paradigms of TCP-friendly congestion control mechanisms have been proposed in the literature [2]–[5], [7], [8]: equation-based rate control and window-based binomial control. Besides the difficulties to accurately measure and estimate the loss event rate and other parameters to calculate the sending rate, equation-based rate control protocols, *e.g.*, TFRC, encounter the similar problem as UDP over wireless links: without ARQ, high transmission error rate may not be tolerable and may result in low flow throughput; with ARQ, the delay outage rate is quite large, since the rate control mechanism does not have the acknowledgment self-clocking property, and it responds to channel variations more slowly than window-based control. Delay performance of TFRC flows over wireless links has been studied in [35].

For window-based control, besides the AIMD mechanism, other mechanisms based on binomial congestion control have been proposed [5], *e.g.*, Inverse Increase and Additive Decrease (IIAD), Square-root Increase and Square-root Decrease (SQRT), which increase or decrease *cwnd* more smoothly than AIMD. Although other binomial congestion controlled flows can be TCP-friendly under certain circumstances, it is very difficult if not impossible for them to achieve TCP-friendliness independent of the bottleneck link capacity [8]. Therefore, we choose the AIMD window-based control mechanism which can guarantee TCP-friendliness no matter what the bottleneck link capacity is and how many flows coexist in the link.

Using *rwnd* to enhance TCP performance has been proposed in the literature [12], [36], [37]. In [12], *rwnd* is used to enhance fairness and reduce packet losses in the wired link. In our approach, we set the optimal *rwnd* not only to efficiently utilize the time-varying wireless link and avoid buffer overflow at the BS, but also to bound the delay outage rate. In [36] and [37], *rwnd* is used to enhance TCP performance for hybrid ATM/IP and third generation wireless/wired networks, by adaptively adjust *rwnd* according to the queue length (or free buffer size) at the interface node. Since the *rwnd* is determined with the assumption that there is no delay and loss in the wired domain, the performance of the schemes in [37] degrades when the *rtt* is above 100 ms or the packet loss rate due to congestion in wired domain exceeds 0.1%. In this paper, instead of frequently changing *rwnd* according to the current queue length, we derive the optimal *rwnd* with consideration of the link profile, flow *rtt*, and application QoS requirements, and the simulation results show the robustness of our approach.

Although we have list some of the work closely related to the transport layer protocol design for multimedia applications over wireless links, the list is not exhaustive, and there are a lot of on-going research in this active area.

VII. Conclusions

The TCP-friendly AIMD protocol has been applied for supporting time-sensitive multimedia applications over hybrid wired/wireless networks. With appropriate protocol parameters, AIMD flows can efficiently utilize wireless resources under the constraint of the delay outage bound. Simulation results have validated our analysis, demonstrated the feasibility of our approach, and shown that the AIMD protocol can outperform the unresponsive UDP protocol when supporting multimedia applications over hybrid networks.

The research presented in this paper has been focused on one-hop infrastructure-based wireless networks. Research on multi-hop wireless ad hoc networks is under investigation.

ACKNOWLEDGMENTS

This work has been supported by a Postgraduate Scholarship and a Strategic Project Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] E. Blanton, M. Allman, K. Fall, and L. Wang, "A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP," *IETF RFC 3517*, 2003.

[2] W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, no. 2, pp. 172–186, 1999.

[3] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Proc. IEEE INFOCOM'99*, March 1999, pp. 1337–1345.

[4] S. Floyd, M. Handle, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM'2000*, 2000, pp. 43–56.

[5] D. Bansal and H. Balakrishnan, "TCP-friendly congestion control for real-time streaming applications," MIT, Tech. Rep. MIT-LCS-TR-806, May 2000.

[6] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control," May 2000, available http://www.aciri.org/tfrc/tcp-friendly.TR.ps.

[7] Y. R. Yang and S. S. Lam, "General AIMD congestion control," University of Texas, Tech. Rep. TR-2000-09, May 2000, available http://www.cs.utexas.edu/users/lam/NRL/TechReports/.

[8] L. Cai, X. Shen, J. Pan, and J. W. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications," *IEEE Transactions on Multimedia*, 2004, to appear. Available http://bbcr.uwaterloo.ca/~cai/tomm-03.pdf.

[9] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," November 2004, work in progress. Available http://www.icir.org/kohler/dcp/draft-ietf-dccp-spec-09.txt.

[10] S. Floyd and E. Kohler, "Profile for DCCP congestion control ID 2: TCP-like congestion control," November 2004, work in progress. Available http://www.icir.org/kohler/dcp/draft-ietf-dccp-ccid2-08.txt.

[11] S. Floyd and S. McCanne, "Network Simulator," LBNL public domain software. Available via ftp from ftp.ee.lbl.gov. NS-2 is available in http://www.isi.edu/nsnam/ns/.

[12] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP rate control," *ACM Computer Communications Review*, vol. 30, no. 1, pp. 45–58, 2000.

[13] F. P. Kelly, "Stochastic models of computer communication systems," *Journal of the Royal Statistical Society*, vol. B47, no. 3, pp. 379–395, 1985.

[14] D. Loguinov and H. Radha, "End-to-end rate-based congestion control: convergence properties and scalability analysis," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 564–577, 2003.

[15] G. Fairhurst and L. Wood, "Advice to link designers on link Automatic Repeat reQuest (ARQ)," *IETF RFC 3366*, 2002.

[16] W. Li, "Overview of fine granularity scalability in MPEG-4 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.

[17] H. M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.

[18] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, pp. 74–93, 2001.

[19] L. Cai, X. Shen, J. W. Mark, and J. Pan, "Performance analysis of AIMD-controlled multimedia flows in wireless/wireline networks," University of Waterloo, E&CE Tech. Rep., Tech. Rep., 2004, available http://bbcr.uwaterloo.ca/~cai/tech-04-1.pdf.

[20] M. Vojnovic, J. L. Boudec, and C. Boutremans, "Global fairness of Additive-Increase and Multiplicative-Decrease with heterogeneous round-trip times," in *Proc. IEEE INFOCOM'00*, vol. 3, 2000, pp. 1303–1312.

[21] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, August 1999.

[22] A. Gurtov, M. Passoja, O. Aalto, and M. Raitola, "Multi-layer protocol tracing in a GPRS network," in *Proc. IEEE VTC'02*, vol. 3, 2002, pp. 1612–1618.

[23] "The PingER project," 2004, available http://www-iepm.slac.stanford.edu/pinger/.

[24] D. Zhang and K. M. Wasserman, "Transmission schemes for time-varying wireless channels with partial state observations," in *Proc. IEEE INFOCOM'02*, vol. 2, 2002, pp. 467–476.

[25] V. Jacobson and M. Karels, "Congestion avoidance and control," in *Proc. ACM SIGCOMM'88*, 1988, pp. 314–329.

[26] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin, "AIRMAIL: A link-layer protocol for wireless networks," *Wireless Networks*, vol. 1, no. 1, pp. 47–69, 1995.

[27] H. M. Chaskar, T. V. Lakshman, and U. Madhow, "TCP over wireless with link level error control: analysis and design methodology," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 605–615, 1999.

[28] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP performance over wireless network at the link layer," *Mobile Networks & Applications*, vol. 5, no. 1, pp. 57–71, 2000.

[29] K. Ramakrishnan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP," *IETF RFC 2481*, January 1999.

[30] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[31] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz, "Improving TCP/IP performance over wireless networks," in *Proc. ACM Mobicom'95*, 1995, pp. 2–11.

[32] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. ICDCS'95*, 1995, pp. 136–143.

[33] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *ACM Comp. Comm. Review*, vol. 27, no. 5, pp. 19–43, 1997.

[34] J. W. Mark, J. Pan, and X. Shen, "Mobility support in hybrid wireless/IP networking," *Elsevier Journal of Computer Communications*, vol. 26, no. 17, pp. 1990–1997, 2003.

[35] H. Shen, L. Cai, and X. Shen, "Performance analysis of equation based TFRC over wireless links with link level ARQ," in *Proc. IEEE Globecom'04*, 2004, available http://bbcr.uwaterloo.ca/~cai/gc-04.pdf.

[36] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit window adaptation: a method to enhance TCP performance," *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, pp. 338–350, 2002.

[37] M. C. Chan and R. Ramjee, "Improving TCP/IP performance over third generation wireless networks," in *Proc. IEEE INFOCOM'04*, 2004.