QoS Support for Multimedia Traffic in Wireless/IP Networks Using AIMD Protocols

Lin Cai, *Student Member, IEEE*
Xuemin (Sherman) Shen*, *Senior Member, IEEE*
Jon W. Mark, *Life Fellow, IEEE,* and
Jianping Pan, *Member, IEEE*

Centre for Wireless Communications
Department of Electrical & Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
{*cai, xshen, jwmark, jpan*}@bbcr.uwaterloo.ca

Correspondent Author: Prof. Xuemin (Sherman) Shen
Department of Electrical & Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
Phone: (519) 8884567 ext. 2691
Fax: (519) 7463077
Email: xshen@bbcr.uwaterloo.ca

Abstract

In this paper, we study how to select the transport layer protocol parameters with considerations of wireless link characteristics and application QoS requirements. In our approach, the inter-layer interactions only exchange protocol parameters between layers. We demonstrate that by appropriately selecting the AIMD protocol parameters, AIMD-controlled flows can efficiently utilize wireless resources and provide satisfactory QoS to time-sensitive multimedia applications. Extensive simulations are performed to validate the analytical results, evaluate the performance, and show that AIMD protocols can outperform the non-responsive UDP protocol when they are used to support multimedia applications over hybrid networks. Since AIMD and TCP protocols share the same congestion control mechanism, AIMD protocols are compatible with the legacy and scalable to be deployed incrementally. In addition, with satisfactory QoS provisioning, end-systems have more incentives to voluntarily regulate multimedia traffic with an AIMD-based congestion controller, which is vital for network stability and integrity.

Index Terms

TCP/IP, AIMD, congestion control, wireless networks, quality of services, network simulation

I. INTRODUCTION

Hybrid wireless/IP networks are anticipated to support multimedia applications and services. In [1], it has been discussed that the dominant transport layer protocol, TCP (Transmission Control Protocol), is not favorable for emerging multimedia applications, which have much tighter and more diverse requirements on delivery timeliness rather than plain object integrity. To support time-sensitive multimedia applications and maintain network stability and integrity, TCP-friendly congestion control has become an active research topic [2]-[11]. Two paradigms of TCP-friendly congestion control mechanisms are proposed in the literature: *equation-based rate control* and *Additive Increase Multiplicative Decrease (AIMD) window control*.

For the equation-based approach, several analytical models [12]-[13] are used to obtain the

long-term TCP throughput as a function of the measured packet[1] loss event rate (p), round-trip time (rtt), Maximum Segment Size (MSS), and initial timeout value (T_0). If these parameters are *readily* available, a rate-controlled sender can regulate its long-term sending rate as a *pseudo* TCP connection does under the same condition. TCP-Friendly Rate Control (TFRC) is a representative scheme in this paradigm [6]. However, the throughput model is quite sensitive to parameters (*e.g.*, p and rtt) that are difficult to measure efficiently and to predict accurately, especially when the flow traverses highly dynamic and error prone wireless links.

On the other hand, window-based AIMD approach inherits the same principle as TCP. Instead of the *increase by one or decrease by half* strategy, AIMD sender increases its congestion window (*cwnd*) by α packets additively when no congestion is sensed; otherwise, it multiplicatively decreases the *cwnd* to β times its previous value. TCP is a special case of AIMD with ($\alpha = 1$, $\beta = 0.5$). Since TCP and AIMD protocols have the same congestion control mechanism, AIMD protocols are compatible with the legacy and scalable to be deployed incrementally. In addition, window-based protocol has the *acknowledgment self-clocking* property, which is particularly useful for traffic regulation when a flow crosses time-varying wireless links. For instance, with the link layer automatic repeat request (ARQ) scheme, when the wireless channel is in deep fading or shadowing condition temporarily, the AIMD sender can immediately reduce the sending rate since the acks are delayed due to low effective throughput in the wireless link. Once the channel condition becomes better, the acks can arrive at the sender at a faster pace, and the sending rate can be increased accordingly. In this paper, we focus on the window-based AIMD protocols and their parameters, in particular for time-sensitive multimedia applications over hybrid wireless/IP networks.

The QoS performance of AIMD-controlled multimedia flows in hybrid networks has been analyzed in [1]. By using token emulation, the queuing delay distribution, the probability of

[1]The terms link *frame*s, network *packet*s and transport *segment*s are used interchangeably in this paper.

delay jitter exceeding a threshold, and the packet loss rate have been derived. In this paper, we investigate how to select protocol parameters to achieve the following design objectives: TCP-friendliness, efficient utilization of wireless resources, and QoS provisioning.

Our main contributions in this paper are twofold. First, we demonstrate that by appropriately selecting the protocol parameters, TCP-friendly AIMD flows can efficiently support time-sensitive multimedia applications over hybrid wireless/IP networks with satisfactory QoS provisioning. The inter-layer interactions in our approach only exchange protocol parameters among application, transport layer protocol, and link layer protocol. Second, by using the publicly-available Network Simulator (*ns-2*) [17], extensive simulations are performed to validate our analysis, demonstrate the feasibility of our proposed approach, and show that AIMD protocols can outperform non-responsive UDP protocol for time-sensitive multimedia applications over hybrid networks.

The remainder of the paper is organized as follows. Section II gives a brief introduction of TCP-friendly AIMD protocols. Based on the analytical results of QoS performance, in Section III, we demonstrate how to set the parameters of the AIMD flows to friendly co-exist with TCP flows, efficiently utilize wireless resources, and statistically guarantee the QoS for time-sensitive multimedia applications. Section IV validates the analytical results and evaluates the performance of AIMD-controlled multimedia flows by extensive simulations, and compares the performance of AIMD protocols and that of non-responsive UDP protocol. Concluding remarks are given in Section V.

II. Brief Introduction of AIMD Protocols

Datagram congestion control protocols are transport layer protocols that implement a congestion-controlled stream of unreliable datagrams [11], which are suitable for time-sensitive multimedia applications. AIMD protocols are window-based datagram congestion control protocols. AIMD sender sends sequenced packets; AIMD receiver sends acknowledgments ($acks$) for correctly

received packets. To avoid a fast sender over-running 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 unacknowledged packets.

AIMD sender does not retransmit lost packets since time-sensitive multimedia applications can tolerate certain degree of packet loss. Therefore, the cumulative acknowledgment scheme used in TCP is not applicable for AIMD protocols. Instead, the AIMD ack has two fields: a 24-bit sequence number, ack_a, which identifies the packet with the largest valid sequence number received from the sender, and an 8-bit selective acknowledgment vector, $sackvec$. The i-th bit in $sackvec$ indicates whether the packet with sequence number $(ack_a - i)$ has been received or not. With $sackvec$, if an AIMD ack on packet $\mathcal{A}$ is lost, the following acks can still inform the AIMD sender that the packet $\mathcal{A}$ has been correctly received. Therefore, the AIMD sender is not required to acknowledge acks, and the AIMD receiver does not retransmit acks.

The congestion control mechanism deployed in AIMD protocols is similar to regular TCP congestion control, except that a pair of parameters, (α, β), is introduced. In specific, 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 value, 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 *cwnd* and halve *ssthresh*, followed by *slow start*. Moderate congestion is indicated by AIMD acks. Normally, when the AIMD acks show that one packet is not received and three packets with higher sequence number are received, the former packet is assumed to be lost, so *cwnd* is reduced by a factor of β (or more precisely, reduced to β times current outstanding data) and

$ssthresh = cwnd$, which is the *exponential backoff* stage.

III. Parameter Selection

To support time-sensitive multimedia applications over hybrid wireless/IP networks, AIMD protocols should appropriately choose the control parameters and the BS should determine the suitable buffer size to achieve the following design objectives: TCP-friendliness, efficient utilization of wireless resources, and QoS provisioning. For AIMD protocols, the configurable control parameters are (α, β) pair and *rwnd*. (α, β) pair controls the increase rate in the *congestion avoidance* stage and the decrease ratio in the *exponential backoff* stage. *rwnd* is originally set to receiver buffer size for flow control purpose. Here, to better support multimedia traffic, we explore if $rwnd$ can be smaller than the receiver buffer size to give a tighter bound of the sender window.

A. TCP-friendliness

Since network stability and integrity depend on congestion control mechanisms voluntarily deployed at end-systems, and TCP is currently the dominant transport layer protocol, the emerging multimedia applications should be regulated by compatible congestion control mechanisms to fairly share network resources with co-existing TCP flows. *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 [14]. It has been shown in [10] that AIMD parameters satisfying the following condition can guarantee TCP-friendliness, no matter what the bottleneck link capacity is:

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

where $0 < \alpha < 3$ and $0 < \beta < 1$. Different applications can choose one of the parameters, and the other parameter is determined by (1).

B. Wireless Link Utilization

The premium resources of the wireless link should be efficiently utilized. On the other hand, with the AIMD mechanism, AIMD flows probe for available bandwidth and overshoot the network capacity frequently, which produce transient congestion and packet losses. For highly multiplexed bottleneck, dynamic probing with AIMD mechanism is necessary since end-systems do not have the knowledge of co-existing traffic in the bottleneck link. However, for a cross-domain connection, the bottleneck is most likely the low multiplexed wireless link. Usually, dedicated wireless channels are allocated to each multimedia user. Although AIMD can achieve good resource utilization in highly multiplexed links, its efficiency is not obvious in low multiplexed ones. In addition, the dynamics of available bandwidth in the wireless domain may not be due to competition of co-existing traffic, but due to the time-varying wireless channel condition. In the following, we study link utilization of the AIMD flow and propose how to appropriately select the protocol parameters to maximize the resource utilization and the flow throughput. First, we consider the simple scenario that an AIMD flow occupies a link with fixed capacity. Then, we study the scenario with a time-varying wireless link.

1) Link with Fixed Capacity: Let an AIMD(α, β) flow occupy a link with fixed capacity. The link buffer size is B packets. The flow's rtt without queuing delay is R slots. We consider the following two cases: (a) $rwnd > R + B$; (b) $rwnd \leq R + B$.

In the first case, since *cwnd* cannot exceed $R + B$, *rwnd* is always larger than *cwnd*, *i.e.*, W equals *cwnd*. As shown in Fig. 1, there are two stages in steady state. In stage I, $cwnd < R$, the queue is empty, and the duration of the rtt equals R slots. In stage II, $cwnd \geq R$, the queue builds up. As *cwnd* increases by α packets per round[2], the queue and the duration of the rtt increases by α packets and α slots per round, respectively. Link utilization is 100% in stage II, since the queue is always non-empty. When the *cwnd* increases to $(R + B)$, there are B packets

[2]A round is defined as the time successfully transmitting and receiving a whole *cwnd* of packets and acks.

in the queue, and packet loss occurs. Consequently, the *cwnd* decreases to $\beta(R+B)$, followed by stage I.

In stage I, the number of packets being transmitted is

$$\sum_{i=0}^{t_I}[\beta(R+B)+\frac{i}{\alpha}]=\frac{R^2-\beta^2(R+B)^2}{2\alpha},$$

where t_I is the number of rounds in stage I, which equals $\left(R-\beta(R+B)\right)/\alpha$. Therefore, the link utilization in stage I, η_I, is given by

$$\eta_I=\begin{cases}\frac{1+\beta}{2}+\frac{\beta B}{2R} & \text{when } B\leq\frac{(1-\beta)R}{\beta},\\ 1 & \text{otherwise.}\end{cases} \tag{2}$$

Equation (2) indicates that for the TCP-friendly AIMD flow with a larger value of β, a smaller size of buffer is required to fully utilize the link capacity. Therefore, it is preferable to choose a large β and set the buffer size to be no less than $\frac{(1-\beta)R}{\beta}$.

In the second case, since $rwnd\leq R+B$, when *cwnd* is less than $(R+B)$, W equals *cwnd*. Since the queue length equals $(W-R)^+<B$, no buffer overflow occurs. When *cwnd* is larger than $(R+B)$, W equals *rwnd*, and the queue length is $(rwnd-R)^+\leq B$. Since no buffer overflow occurs, *cwnd* increases continuously, and W is fixed at *rwnd*. In this case, the link utilization equals $\min(rwnd,R)/R$. Therefore, if $R\leq rwnd\leq(R+B)$, the link can be fully utilized, no matter what the AIMD parameters are.

2) Time-Varying Wireless Link: The wireless channel throughput is time-varying. To fully utilize wireless resources, the channel should not be idle whenever the link layer decides to transmit, *i.e.*, the queue over the wireless link should be non-empty all the time. From Proposition 1 in [1], the sufficient condition to guarantee a non-empty queue is that the window size is no less than R.

Assume that the receiver buffer size is much larger than R, which is generally true. For a cross-domain connection, ideally, if the wireless link is the bottleneck and there is no packet

loss and delay jitter in wireline networks, we can set *rwnd* to R to guarantee that $W = R$ and the queue in the interface node is always non-empty. In reality, packet losses and delay jitter may exist in the wireline networks, and set *rwnd* to R may not be able to absorb delay jitters in the wireline domain, so the wireless channel may be idle sometimes. For instance, when one packet is delayed one more slot in the wireline domain, the queue will be empty for one slot if $rwnd = R$, and the queue can be nonempty all the time if $rwnd = R + 1$. Therefore, it is desirable to set *rwnd* to be $R + \delta$. On the other hand, larger *rwnd* leads to a larger queue length and queuing delay. Therefore, the optimal value of δ should be determined according to the maximum delay jitter in the wireline networks and the delay requirement of the application. The latter will be discussed in Section III-C.

On the other hand, it has been shown in [16] that in a general case, nothing slower than exponential backoff can guarantee network stability when the end-systems have no complete knowledge of the global traffic. The AIMD sender takes any packet loss in the network as congestion signal, and it exponentially decreases its W when congestion signal is captured. Since the throughput of wireless link is time-varying, when $W > B$, packet loss due to buffer overflow may occur, *e.g.*, when the wireless channel has been in the bad condition for a while and no transmission succeeds during that time period. Packet loss due to bad channel condition can trigger the AIMD sender to exponentially reduce its W. However, when channel condition becomes better later, there might not be enough packets for transmission over the wireless link due to the small window size. Thus, to fully utilize the time-varying wireless link and maximize the throughput of the AIMD flow, the wireless link buffer size, B, can be conservatively set to *rwnd*, which is equal to $R+\delta$, to avoid buffer overflow due to wireless channel dynamics; thus, the exponential backoff will not be triggered unnecessarily.

Similarly, when N AIMD flows share the same wireless link, and their minimal rtts satisfy $R_1 \leq R_2 \leq ... \leq R_N$, we can set the sum of all *rwnd*s to be $R_N + \delta$, and set the link buffer size to the sum of all *rwnd*s, in order to fully utilize wireless resources.

C. QoS Provisioning

For time-sensitive multimedia applications, packets that have suffered excessive delay are useless and should be discarded by the receiver. The rtt contains a deterministic part (R) and a random part, and the latter is called delay jitter (d_q). Delay outage probability is defined as the probability that delay jitter (d_q) of a packet is larger than a threshold D. The transmission of a packet which has suffered excessive delay wastes wireless resources, therefore, the delay outage probability should be bounded.

The other QoS concerns for time-sensitive multimedia applications are goodput and packet loss rate. Goodput measures the packets being received with tolerable delay, and can be calculated from flow throughput and delay outage probability. The flow throughput can be maximized when the allocated wireless resources are fully utilized. For multimedia applications, packet loss rate measures the ratio of packets failed to arrive at the multimedia receiver timely. Since bursty losses are particularly undesirable, packet loss rate in a window, $r(W)$, should be bounded. Since local ARQ scheme can inhibit most transmission errors in the wireless link, and the BS buffer size, B, is set to *rwnd* to avoid buffer overflow, for time-sensitive multimedia applications, packet loss rate is approximately the delay outage probability, *i.e.*, $r(W) \approx \Pr\{d_q > D|W\}$. In summary, the main QoS indexes for time-sensitive multimedia applications are flow throughput and delay outage probability.

$\Pr\{d_q > D|W\}$ has been derived in [1]:

$$\Pr\{d_q > D|W\} = \sum_{x=0}^{W-1} T_{R+D}(x). \tag{3}$$

For W, there is a strict increase in delay outage probability. The maximum W satisfying the desired delay outage probability, $W_{\max}$, gives the upper bound of the sender window. For AIMD flows, since the maximum sender window size can be bounded by *rwnd*, *rwnd* can be set to $W_{\max}$ to bound the delay outage probability and the packet loss rate.

In summary, with larger *rwnd* ($rwnd \leq R + B$), the queue at the BS can better absorb the delay jitter in wireline domain, so wireless resources can be fully utilized and the flow throughput can be maximized, as shown in Section III-B. On the other hand, to guarantee the delay outage probability and the packet loss rate, *rwnd* should be no larger than $W_{\max}$. Therefore, the optimal *rwnd* can be set to $W_{\max}$.

Similarly, for multiple AIMD flows in a wireless link, to assign a ratio of wireless bandwidth to AIMD flows with the same or different minimal rtts, the *rwnd*s should satisfy the following equation:

$$rwnd_i \;=\; \frac{p_i}{p_j} rwnd_j + p_i(\mathrm{E}[T_{R_i-1}] - \mathrm{E}[T_{R_j-1}]), \qquad \text{for} \;\; 1 \leq i, j \leq N \tag{4}$$

where p_i and R_i are the i-th flow's bandwidth ratio and minimal rtt, respectively, and $\mathrm{E}[T_{R_i-1}]$ is the mean number of packets being successfully transmitted in $(R_i - 1)$ slots over the wireless link. Under the constraint of (4), optimal *rwnd*s can be determined to bound the delay outage probability and maximize the flow throughputs according to the analytical results in [1].

D. Parameter Selection Procedures

The parameter selection procedure is listed as follows.

1) The time-sensitive multimedia 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) In the transport layer, given β, the AIMD sender calculates α according to the TCP-friendly condition.
3) According to the wireless link characteristics and the physical layer protocol, the BS resource allocation module (in the link layer) calculates the 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.

4) The BS resource allocation module estimates the minimal rtt of the flow, and calculates $W_{\max}$ and B according to the QoS requirements.
5) The BS informs the AIMD sender (in the transport layer) to set *rwnd* equal to $W_{\max}$, assuming that the receiver buffer size is larger than $W_{\max}$.

The computations involved in steps 2) and 3) are very light. The calculations in step 4) can be done offline and the results can be stored in a table, so the BS can look up the table produced offline to determine appropriate *rwnd* and B. Therefore, the parameter selection procedure can be done efficiently during connection establishment phase.

To efficiently support multimedia applications with heterogeneous QoS requirements in hybrid wireless/IP networks, inter-layer interactions are necessary. Nevertheless, such interactions should be minimized for scalable system design purpose. As shown in the above procedure, the inter-layer interactions introduced by our approach only exchange QoS and protocol parameters between application, transport layer protocol, and link layer protocol. Since the effectiveness of TCP congestion control has been demonstrated by the success of the Internet in the past two decades, and the fundamental congestion control mechanism deployed in AIMD protocols is the same as that in TCP, AIMD protocols can achieve the same effectiveness in large-scale networks, and they are also scalable and feasible to be incrementally deployed in hybrid networks.

IV. Simulation Results

To verify the QoS analysis, examine link utilization, and evaluate performance of AIMD protocols, we have performed extensive simulations by using the Network Simulator (*ns-2*) [17]. The simulation topology is shown in Fig. 2. For the target AIMD flow, the sender is at the correspondent host (CH), and the receiver is at the mobile host (MH). The MH and base station (BS) are connected to routers r_1 and r_2, respectively. Cross traffic connections share the r_1r_2 backbone link. The following parameters are used in the simulation unless otherwise explicitly stated. Links between CH, r_1, r_2, and BS are duplex with 100 Mbps. Both r_1 and r_2 are

Random Early Detection (RED) capable. The downlink and uplink bandwidth between BS and MH are 200 Kbps and 100 Kbps, respectively. The buffer size at the BS 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 M-state Markov model. The TCP-friendly AIMD parameters are $\alpha = 0.2$ 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. In the wireless link layer, *persistent* transmission scheme and *optimal* transmission scheme are used when wireless link memory λ is 0 and 0.35, respectively.

A. Delay Outage Rate

To verify the analytical results, we repeat simulations with different values of *rwnd* and record the maximum *rwnd* 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 separately. In the light cross traffic case, the number of cross TCP flows sharing the backbone link between r_1 and r_2 is constrained so that the bottleneck for the target AIMD flow is always the wireless link; in the heavy cross traffic case, the number of co-existing TCP flows is time-varying so that the bottleneck is the backbone link between 30 s and 60 s.

With the *persistent* transmission scheme, Figs. 3(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. Since the *persistent* transmission scheme is suitable for memoryless channel, here, the wireless channel condition evolves following a two state Markov model with channel memory equal to zero, and the average packet error rates, (p_e), is from 1% to 10%. The local retransmission delay is 5 ms.

The analytical and simulation results with the *optimal* transmission scheme are shown in Figs. 4(a) and (b). The wireless channel evolves following a three-state Markov model with channel memory equal to 0.35. According to the *optimal* transmission scheme in [15], the BS suspends transmission for one slot if the previous transmission fails. There is no local retransmission delay for the *optimal* transmission scheme (*e.g.*, in a TDMA system, each link is assigned certain time slots, and the time duration between two consecutive slots of the link is long enough to get the local acknowledgment).

Comparing Figs. 3 and 4, with the *optimal* transmission scheme, the maximum *rwnd* to guarantee delay outage rate is smaller than that with the *persistent* transmission scheme. This is because the *optimal* transmission scheme makes the following tradeoff — achieving multi-user gain and higher power efficiency at the cost of possible lower throughput of a particular flow. In our simulation, only a single wireless link is simulated, and the interferences between wireless links are not addressed. Therefore, the multi-user gain is not reflected in the simulation results, and the throughput with the *optimal* transmission scheme is less than that with the *persistent* transmission scheme. Since our focus is on the performance of the transport protocol, we do not explore the link layer optimization problem here. Nevertheless, given the link layer transmission scheme, the BS can calculate the link layer throughput distribution, and select the suitable protocol parameters for the transport layer protocol. Figs. 3 and 4 also show that the analytical results match well with the simulation results when the cross traffic is light, and the analytical ones are slightly more conservative when the cross traffic is heavy.

In summary, no matter whether the bottleneck link is the wireless link or not, and no matter what link layer transmission scheme is used, it is feasible to determine a suitable *rwnd* beforehand to bound the delay outage rate. Therefore, we only present simulation results with *persistent* transmission scheme over memoryless wireless channel in the remaining of this paper due to the space limitation.

B. Link Utilization and TCP-friendliness

Ideally, for flow/congestion control in hybrid wireless/IP networks, the allocated wireless resources should be fully utilized if the bottleneck is the wireless link, and the cross-domain AIMD flow should only occupy its fair share of bandwidth when the bottleneck is a highly multiplexed backbone link.

To examine link utilization and TCP-friendliness of AIMD protocols, the number of co-existing TCP flows in the backbone link is changed from 0 to 100. The packet size of background TCP flows is 1250 bytes. Normalized throughput is defined as the number of packets being received successfully per time slot. Fig. 5 plots the normalized throughput for the target AIMD flow and the average normalized throughput of all co-existing TCP and AIMD flows, with the average packet error rates equal to 0.1. When the number of co-existing flows is less than 50, the average normalized throughput in the backbone link is larger than 1, and the bottleneck is the wireless link. Therefore, the normalized throughput of the target AIMD flow should be $(1-p_e)$ packet per slot to fully utilize the wireless link. When the number of co-existing 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 to be TCP-friendly. Fig. 5 shows that the target AIMD flow can efficiently utilize the allocated wireless resources when the bottleneck link is the wireless link, and can friendly co-exist with TCP flows when the bottleneck is the highly multiplexed backbone link. The simulation results with other p_e show the same tendency.

It is noticed that when the bottleneck link of the target flow is in the wireline 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. Alternatively, the BS can let a number of AIMD flows share one link to achieve statistical multiplexing gain, and this will be evaluated in the following subsection.

C. Multiple AIMD Flows

Let two AIMD flows have the same share of wireless resources. Their *rwnd*s are set according to (4) and delay outage bound, *i.e.,* $\Pr\{d_q > D|\boldsymbol{W}\} \leq 0.01$. The link buffer size is set to the sum of all *rwnd*s to avoid buffer overflow at the BS. Fig. 6(a) shows the normalized throughputs and delay outage rates of two flows with the same minimal rtts of 17 slots, with respect to the average packet error rate p_e. In Fig. 6(b), two AIMD flows have different minimal rtts of 17 slots and 25 slots, respectively. The simulation results demonstrate that the co-existing AIMD flows can fairly share the wireless link, and satisfy the delay outage bounds, no matter whether they have the same minimal rtts or not.

The same conclusion can be drawn when the number of flows sharing the wireless link is increased. Figs. 7(a) and (b) show the normalized throughputs and the delay outage rates for four co-existing AIMD flows. These four flows are designed to occupy the same share of wireless resources. In Fig. 7(a), all four flows have the same minimal rtts of 17 slots; in Fig. 7(b), flow 1 and flow 3 have the same minimal rtts of 17 slots, and flow 2 and flow 4 have the same minimal rtts of 25 slots. Since *rwnd* is an integer, we may not be able to get a group of *rwnd*s to satisfy (4) exactly. Therefore, in different rtts cases, the results slightly deviate from our designed target, *e.g.,* in some case the throughputs of flows have 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 the selfish users' point of view, they would like to discard any congestion control in their systems if such control has negative effects on their perspective QoS. How to punish the greedy or malice is beyond the scope of this paper. However, we note that by appropriately choosing the protocol parameters,

responsive AIMD protocols can outperform unresponsive UDP when they are used to support multimedia applications in error-prone wireless networks. This can be an incentive for end-systems to deploy AIMD congestion control.

Since the UDP has no closed-loop control mechanism, the sender just keeps on sending at the source rate. Assume that the UDP sender still has the knowledge of the wireless link, *i.e.*, link bandwidth, p_e, *etc.*. Then, the UDP sender can determine an optimal sending rate accordingly. For the error-prone wireless link, the BS can choose to use or not to use the local ARQ for UDP traffic. If without ARQ, packet losses due to transmission error is approximately p_e, which might be too severe when the channel condition is poor. If with ARQ, packet losses due to transmission errors can be reduced significantly. However, unlike the AIMD sender which can slow down the transmission when the channel is in the bad condition due to its *acknowledgment self-clocking* property, the UDP sender will not change the sending rate. Consequently, the queue is built up at the BS, and the queuing delay and delay outage probability increase quickly.

In the simulations, the maximum tolerable delay jitter is 10 slots. To avoid excessive delay outage rate, BS buffer size is set to 10 packets for UDP flows. 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 link level ARQ, and the BS buffer size is set to *rwnd*. Normalized goodput is defined as the number of packets being successfully received within the delay bound per slot, which is equivalent to the normalized throughput minus the delay outage rate.

Fig. 8 compares the normalized goodput and packet loss rate for UDP (with and without ARQ) and AIMD, with p_e equal to 0.1. Simulation results show that, without ARQ, the packet loss rate for the UDP flow is approximately 0.1; this makes reconstruction of multimedia streams at the receiver more difficult. When local ARQ is deployed, the source rate of the UDP flow should be less than the average wireless link throughput to avoid excessive packet loss rate. As shown in fig. 8, no matter how the UDP sender adjusts its sending rate, and no matter whether the wireless link deploys the local ARQ or not, the goodputs of the UDP flows are consistently

lower than that of the AIMD flow, and the packet loss rates of the UDP flows are consistently higher than that of the AIMD flow.

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

V. Conclusions

In this paper, we have shown that by appropriately selecting the protocol parameters for TCP-friendly AIMD flows, wireless resources can be efficiently utilized and the QoS requirements such as the packet loss rate and the delay outage probability can be statistically guaranteed for time-sensitive multimedia applications. Simulation results have validated our analysis, demonstrated the feasibility of our proposed approach, and shown that AIMD protocols can outperform the non-responsive UDP protocol when they are used to support multimedia applications over hybrid networks.

Acknowledgment

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] L. Cai, X. Shen, J. W. Mark, and J. Pan, “Performance Analysis of AIMD-Controlled Multimedia Flows in Wireless/IP Networks,” *University of Waterloo, E&CE Tech. Rep.*, May 2004. [Online]. Available: http://bbcr.uwaterloo.ca/ cai/tech-04-1.pdf

[2] W. Tan and A. Zakhor, “Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol,” *IEEE Trans. on Multimedia*, 1(2):172–186, 1999.

[3] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, “A model based TCP-Friendly rate control protocol,” *University of Massachusetts, CMPSCI Tech. Rep. TR 98-04*, 1998.

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

[5] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, “TCP rate control,” *ACM Computer Communications Review*, 30(1):45–58, 2000.

[6] S. Floyd and M. Handle, J. Padhye, and J. Widmer, “Equation-based congestion control for unicast applications,” *Proc. ACM SIGCOMM’2000*, pp. 43–56, 2000.

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

[8] I. Rhee, V. Ozdemir, and Y. Yi, “Tear: TCP emulation at receivers - flow control for multimedia streaming,” *North Carolina State University, Tech. Rep.*, April 2000.

[9] Y. R. Yang and S. S. Lam, “General AIMD congestion control,” *University of Texas, Tech. Rep. TR-2000-09*, May 9, 2000. [Online]. Available: http://www.cs.utexas.edu/users/lam/NRL/TechReports/

[10] L. Cai, X. Shen, J. Pan, and J. W. Mark, “Performance analysis of TCP-friendly AIMD algorithms for multimedia applications,” *IEEE Trans. on Multimedia*, to appear, 2004.

[11] E. Kohler, M. Handley, S. Floyd, and J. Padhye, “Datagram Congestion Control Protocol (DCCP)” Feb. 2004. [Online]. Available: http://www.icir.org/kohler/dcp/draft-ietf-dccp-spec-06.txt

[12] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, “The macroscopic behavior of the TCP congestion avoidance algorithm,” *ACM Computer Communication Review*, 27(3), 1997.

[13] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” *Proc. ACM SIGCOMM’98*, pp. 303–314, 1998.

[14] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Trans. on Networking*, 7(4):458–472, 1999.

[15] D. Zhang and K. M. Wasserman, “Transmission schemes for time-varying wireless channels with partial state observations,” *Proc. IEEE Infocom’02*, 2:467–476, 2002.

[16] F. P. Kelly, “Stochastic models of computer communication systems,” *Journal of the Royal Statistical Society B 47*, 3:379–395, 1985.

[17] 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/

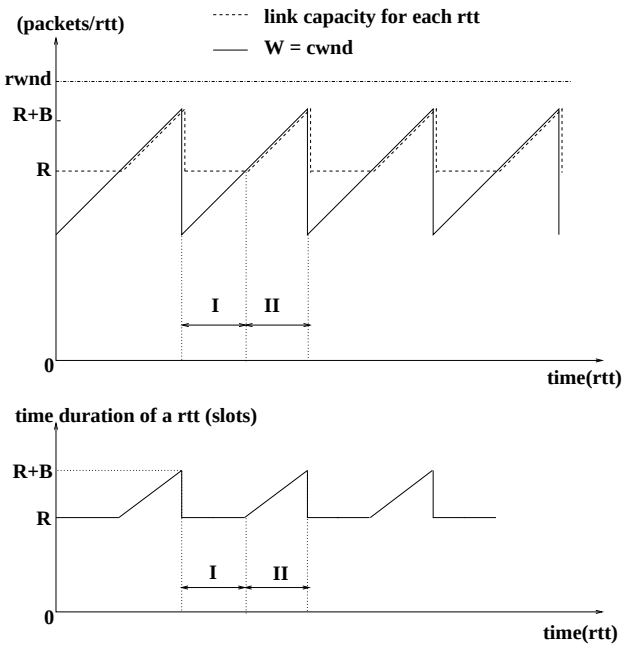


Fig. 1. Window trace, $rwnd > R + B$

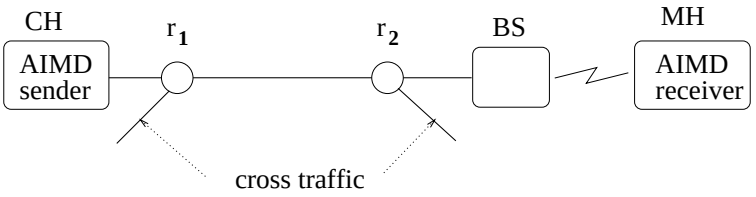
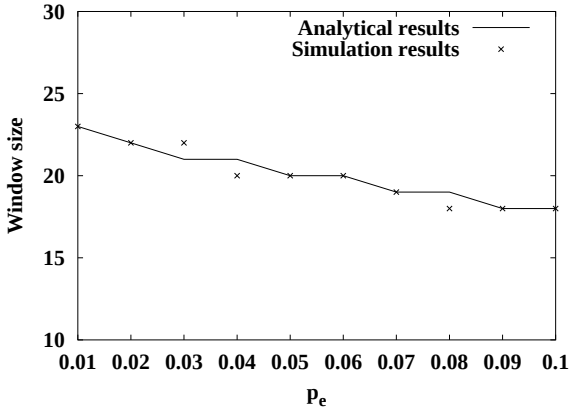
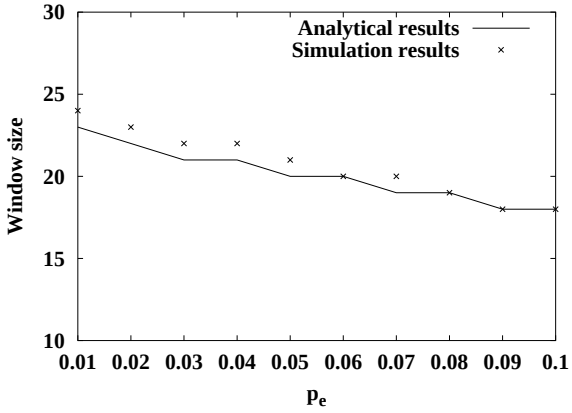


Fig. 2. Simulation topology

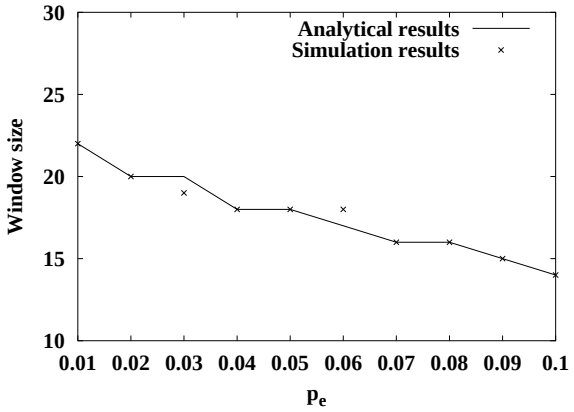


(a) light cross traffic

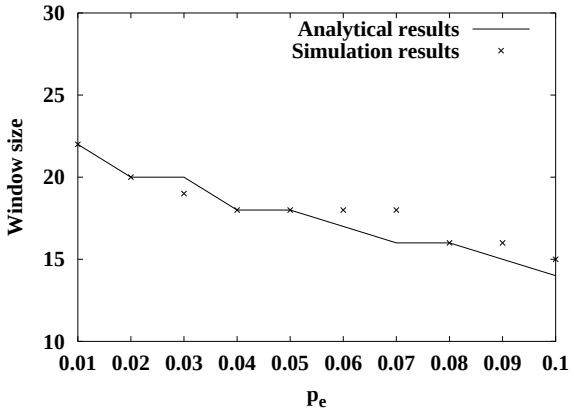


(b) heavy cross traffic

Fig. 3. Maximum *rwnd* to guarantee delay outage rate, *persistent* tansmission scheme



(a) light cross traffic



(b) heavy cross traffic

Fig. 4. Maximum *rwnd* to guarantee delay outage rate, *optimal* transmission scheme

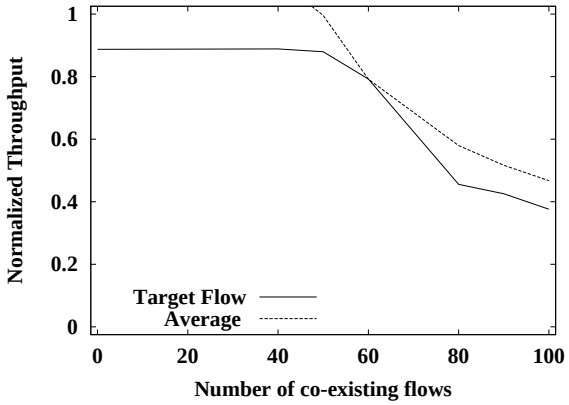
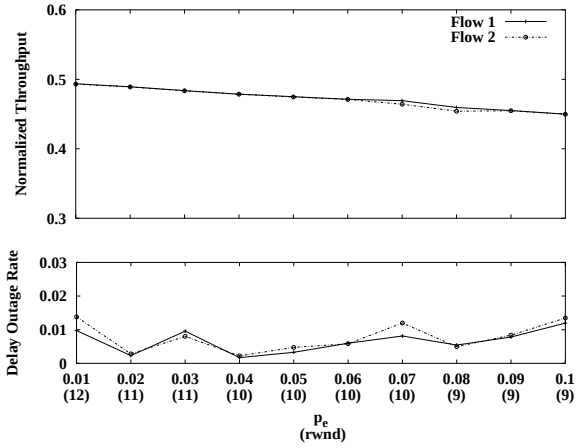
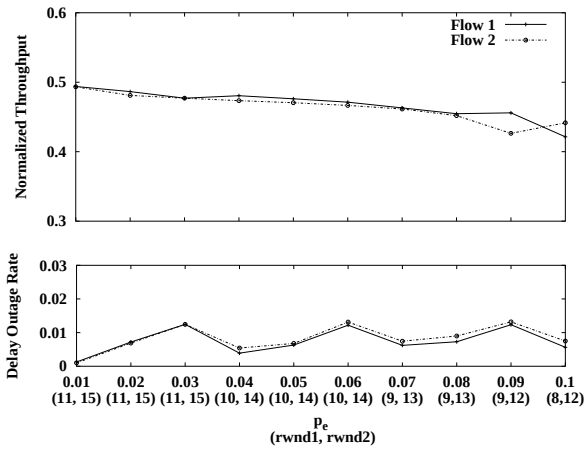


Fig. 5. Normalized throughput, $p_e = 0.1$

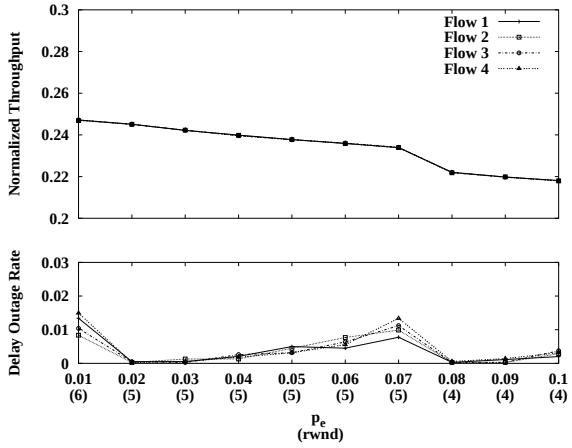


(a) same rtts

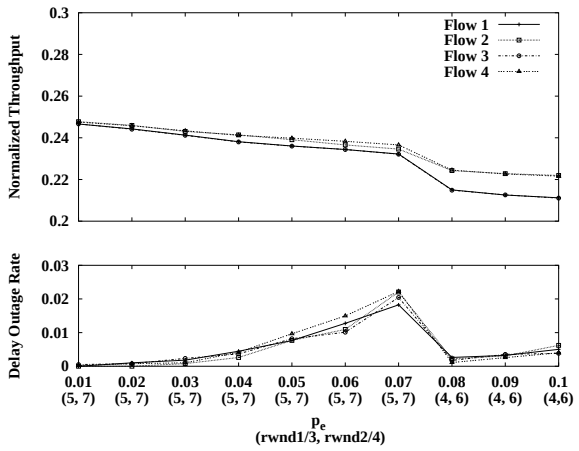


(b) different rtts

Fig. 6. Two AIMD flows sharing the wireless link



(a) same rtts



(b) different rtts

Fig. 7. Four AIMD flows sharing the wireless link

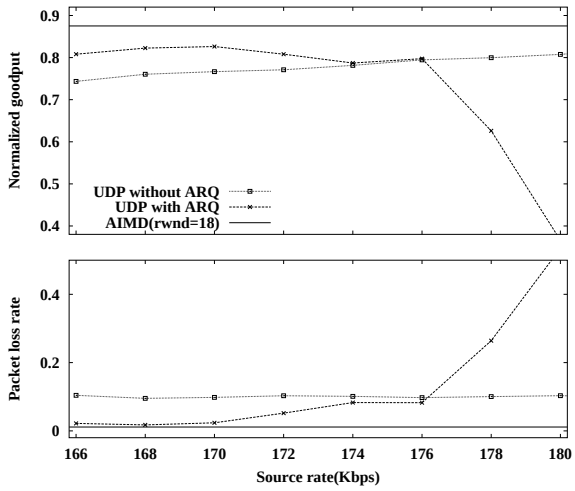


Fig. 8. AIMD vs. UDP, $p_e = 0.1$