QoS Support in Wireless/Wired Networks Using the TCP-Friendly AIMD Protocol

Lin Cai, Xuemin (Sherman) Shen, Jon W. Mark, and Jianping Pan

Abstract— **We propose a TCP-friendly Additive Increase and Multiplicative Decrease (AIMD) based Datagram Congestion Control Protocol (DCCP) protocol for supporting multimedia traffic in hybrid wireless/wired networks. We further demonstrate how to select the protocol parameters to fairly and efficiently utilize network resources with the consideration of quality of service (QoS) requirements. Since the parameter selection procedure 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 layer protocol and the layered structure of the Internet. Extensive simulations are performed to evaluate the proposed protocol. It is shown that the AIMD protocol can appropriately regulate multimedia traffic to efficiently utilize the wireless link and fairly share the network resources with coexisting TCP flows, and it can provide satisfactory QoS for delay-sensitive multimedia applications. In addition, AIMD protocol can outperform the non-responsive User Datagram Protocol (UDP) when transporting multimedia traffic over hybrid wireless/wired networks. 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, integrity, and future proliferation.**

Index Terms— **Protocols, wireless networks, quality of services, congestion control, simulation.**

I. Introduction

Wireless cellular systems and the Internet are expected to converge to an all-IP information transport infrastructure, allowing users to access the hybrid networks for multimedia services anywhere, anytime. However, the Internet and wireless cellular networks have different resource management approaches. In the Internet, under the control of the transport layer protocol, coexisting flows can fairly share network resources in a distributed manner. In wireless cellular networks, due to limited wireless bandwidth, centralized resource management and allocation schemes are used to maintain the QoS of existing and handoff calls. Therefore, for cross-domain traffic, transport layer protocols should not only regulate the traffic to fairly share the highly multiplexed wired links in a distributed manner, but also efficiently utilize the lightly multiplexed or dedicated wireless links.

On the other hand, hybrid networks are anticipated to provide multimedia services; therefore, transport layer protocols should also provide satisfactory quality of services to heterogeneous multimedia applications. However, the currently dominant transport layer protocol, TCP[1], is not favorable for emerging multimedia applications that have much tighter and more diverse requirements on delivery timeliness rather than plain object integrity. TCP uses a congestion window (*cwnd*) to probe for available bandwidth and respond to network congestion, according to the AIMD congestion control mechanism: the 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* strategy, even an adaptive and scalable source coding scheme cannot hide the flow throughput fluctuation; thus, the user-perceived multimedia quality may change drastically. In addition, TCP offers a reliable data transfer service, so the sender will retransmit corrupt or lost packets. End-to-end retransmissions may introduce intolerable delay and delay jitter for delay-sensitive multimedia applications.

Recently, TCP-friendly congestion control for multimedia applications has become an active research topic [2]–[7]. Two paradigms of TCP-friendly congestion control mechanisms have been proposed in the literature: *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 with preferred timely delivery service to fully reliable service, the Datagram Congestion Control Protocol (DCCP) has been proposed, which implements a congestion-controlled unreliable flow of datagrams [9]. DCCP can choose 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. Delay guarantee for delay-sensitive applications over wireless networks is especially challenging, due to time-varying and error-prone wireless channel characteristics.

In this paper, we propose an AIMD-based DCCP protocol (named AIMD protocol), which is extended from the DCCP protocol with TCP-like congestion control (DCCP-2) [10]. The AIMD protocol inherits the same congestion control mechanism as that of TCP, so it is compatible with the legacy and scalable to be deployed incrementally. Instead of the increase-by-one or decrease-by-half strategy, AIMD sender increases its *cwnd* by α segment when no congestion is sensed; otherwise, it decreases the *cwnd* to β times its previous value [6], [7]. The (α, β) pair can be flexibly chosen by applications, under the constraint of the TCP-friendly condition derived in [7]. To support heterogeneous multimedia applications over hybrid wireless/wired networks, we study how to appropriately select the protocol parameters according to the QoS requirements

Manuscript received July 30, 2004; revised February 12, 2005; accepted February 16, 2005. The associate editor coordinating the review of this paper and approving it for publication was W. Liao.

L. Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6, Canada (email: cai@uvic.ca).

X. Shen and J. W. Mark are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: xshen, jwmark@bbcr.uwaterloo.ca).

J. Pan is with the Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: pan@uvic.ca).

Digital Object Identifier 10.1109/TWC.2006.02026.

[1] In this paper, the acronym TCP refers to TCP SACK [1]

and the wireless channel profile. By evaluating the QoS performance of AIMD-controlled flows in hybrid networks, the protocol parameters are chosen with the consideration of the following design objectives: TCP-friendliness, efficient resource utilization, and QoS provisioning.

The main contributions of this paper are as follows. First, an unreliable AIMD protocol is proposed for delay-sensitive multimedia applications. Second, we demonstrate how to select the protocol parameters, such that the TCP-friendly AIMD protocol can efficiently support delay-sensitive applications over hybrid wireless/wired networks with satisfactory QoS provisioning. Since the parameter selection procedure 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 layer 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 wireless links with different channel profiles and physical and link layer protocols. Furthermore, by using the Network Simulator (NS-2) [11], extensive simulations have been performed to verify the feasibility and advantages of our approach. Simulation results show that the AIMD protocol can outperform the non-responsive UDP protocol for delay-sensitive multimedia applications over hybrid networks.

The remainder of the paper is organized as follows. Section II introduces the AIMD protocol. The system model and QoS performance indexes are given in Section III. Based on the performance analysis, the parameter selection procedure is proposed in Section IV. In Section V, simulation results are given to validate the analytical results, evaluate the performance of AIMD-controlled multimedia flows, and compare the performance of the AIMD protocol with that of the TCP and UDP protocol. Related work is given 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 to regulate delay-sensitive multimedia traffic. The AIMD sender sends packets[2] whose sequence numbers are in the range of a sliding window, which is called the sender window; AIMD receiver sends acknowledgments (*ack*s) for correctly received packets. The sender window size, W, is adjusted according to the flow and congestion control mechanism. In the following, we focus on the two main components of the AIMD protocol: the acknowledgment scheme and the flow and congestion control mechanism. 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. On the other hand, the acknowledgment scheme of DCCP-2 requires the sender to acknowledge the receiver's acknowledgments, which is too complicated to implement [12]. Here, we present an applicable 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 (SackVec) with negotiable length. ack_a identifies the packet with a valid sequence number (in circular sequence space) received from the sender; the i-th bit in SackVec indicates the status (whether or not a packet has been received) of the packet with sequence number $(ack_a - i)$. The length of SackVec is denoted as $|SackVec|$.

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

B. Flow and Congestion Control

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

AIMD Receiver — The algorithm used at the AIMD receiver side is shown in Fig. 1. Let $Rwnd_{left}$ and $Rwnd_{right}$ denote the left and right edge of the receiver window, respectively.

When the AIMD receiver receives a packet with sequence number s,
1) if s is to the left of $Rwnd_{left}$ ($s < Rwnd_{left}$), the packet is discarded without any *ack*;
2) if s lies within the receiver's window ($Rwnd_{left} \leq s \leq Rwnd_{right}$), the received packet is buffered, and the receiver sends an *ack* which shows that the packet s is received and also indicates the status of the packets with sequence number between $s - |SackVec|$ and $s - 1$;
3) if s is larger than $Rwnd_{right}$, the packet is buffered, and the receiver sends an *ack* to the sender, advances $Rwnd_{right}$ to s (if $Rwnd_{left} < s - rwnd + 1$, $Rwnd_{left}$ is advanced to $s - rwnd + 1$). Therefore, the buffered packets always have the *largest* sequence numbers (*largest* is measured in circular sequence space[3]), which is desired for delay-sensitive applications.

When the multimedia application fetches n packets from the AIMD receiver buffer, the packets with the sequence number between $Rwnd_{left}$ and $Rwnd_{left} + n - 1$ are delivered to the application, and $Rwnd_{left}$ is advanced by n.

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

[3] To compare two sequence numbers a and b in circular sequence space, if $0 < b - a < 2^{23}$ or $b - a < -2^{23}$, b is larger than a; if $0 < a - b < 2^{23}$ or $a - b < -2^{23}$, b is smaller than a.

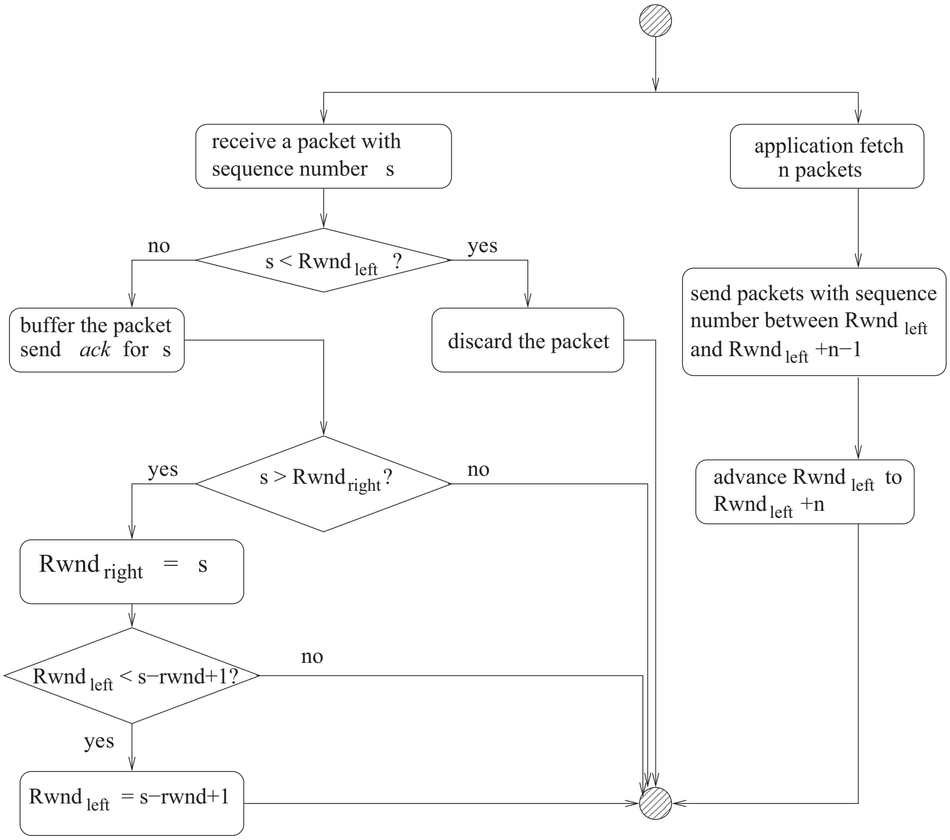

receive a packet with sequence number s
application fetch n packets
no
s < Rwnd left ?
yes
buffer the packet send ack for s
discard the packet
send packets with sequence number between Rwnd left and Rwnd left +n−1
yes
s > Rwnd right ?
no
advance Rwnd left to Rwnd left +n
Rwnd right = s
no
Rwnd left < s−rwnd+1?
yes
Rwnd left = s−rwnd+1

Fig. 1. AIMD receiver.

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 used. 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 the sender window (W) is the minimum of *cwnd* and *rwnd*.

The *cwnd* evolves in three phases. Initially, or 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* phase. 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* phase, until eventually congestion occurs. Severe congestion is indicated by timeout, which forces the AIMD sender to set the *cwnd* to *IW* and halve the *ssthresh*, followed by slow start. Moderate congestion is indicated when the AIMD *ack*s show that one packet is not received and three or more packets with larger sequence number 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 to β times the number of currently outstanding packets) and $ssthresh = cwnd$, which is the *exponential backoff* phase. Similar to TCP, the AIMD sender will backoff only once for all packet losses within one window.

AIMD Sender — The flow and congestion control algorithm at the AIMD sender side is shown in Fig. 2. Specifically, if timeout occurs, the AIMD sender re-initials the *cwnd* and W, advances the left edge of the sender window (W_{left}), and sends W new packets (event **A** in the figure); 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 $W_{left}, \cdots, W_{left} + j - 1$ are received successfully, the *cwnd* is enlarged by j if in the slow start phase (event **B**), or enlarged by $\alpha j/cwnd$ if in the congestion avoidance phase (event **C**). If the *ack* indicates that k out-of-order packets with sequence numbers larger than W_{left} 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_{left} is advanced to guarantee that the window is reduced only once per *rtt* (event **E**). If ack_a is less than W_{left}, 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 packets at a certain time[4].

C. Advantages of Window-Based AIMD Mechanism

The 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) in general, anything slower than exponential backoff cannot guarantee network stability when the end

[4] The packet spacing technology has been deployed in Rate Adaptation Protocol (RAP) [3] and TCP rate control [13].

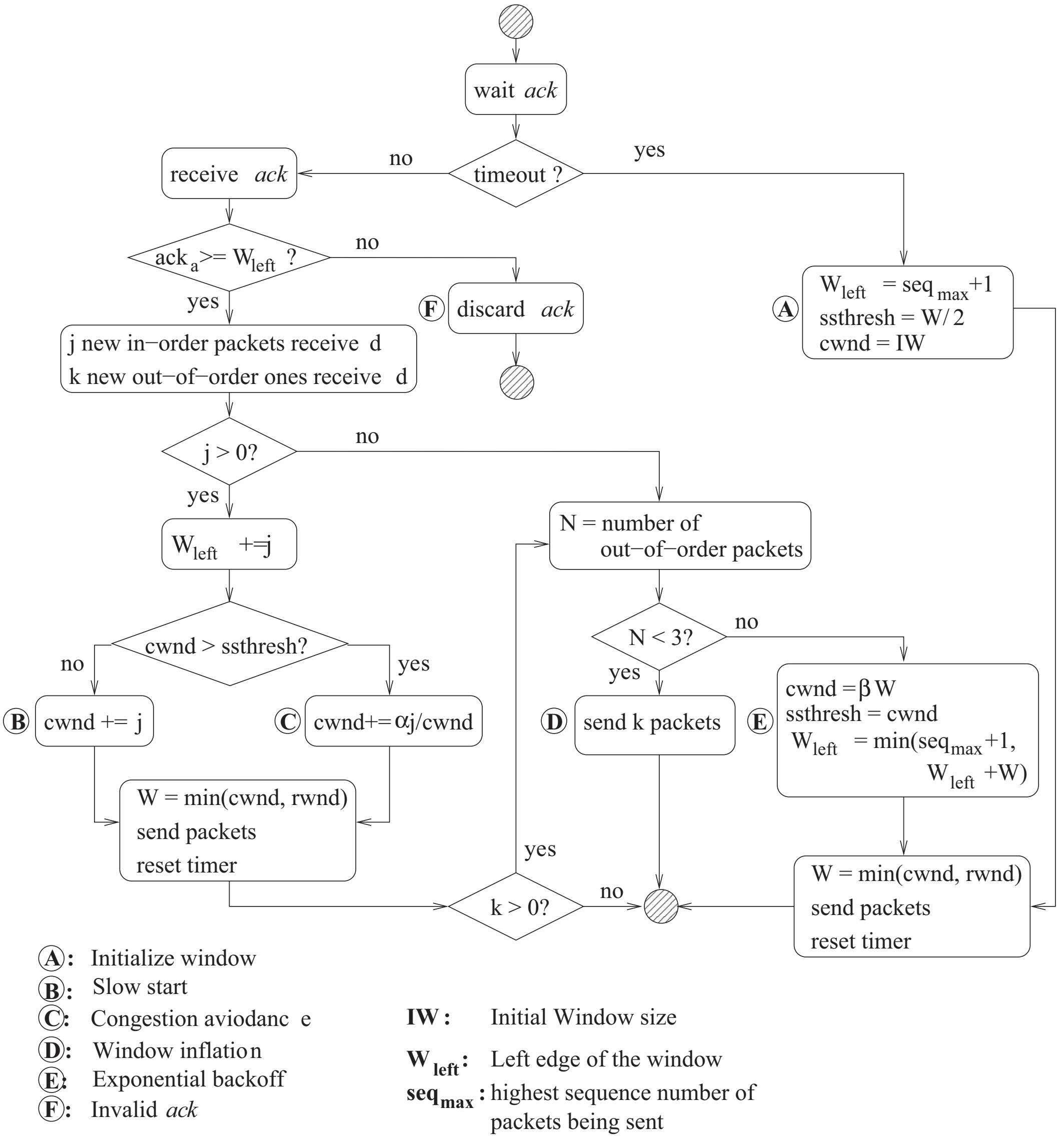


Fig. 2. AIMD sender.

systems have no complete knowledge of the global traffic [14]; c) AIMD is the *only* TCP-friendly binomial control with monotonic convergence to fairness [15], and under the TCP-friendly condition derived in [7], 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.

In addition, window-based protocols have the *acknowledgment self-clocking* property, which is particularly useful to regulate traffic over time-varying wireless links. With the link-level Automatic Repeat reQuest (ARQ) and/or Channel-State-Dependent (CSD) packet transmission schemes, when the wireless channel is in a bad 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, the queue length at the wireless link is always constrained by the window size.

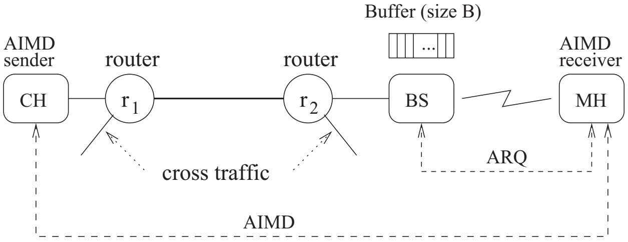


Fig. 3. System model.

III. System Model

We consider the scenario shown in Fig. 3. Let a cross-domain 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 multimedia connection is regulated by the AIMD(α, β)

protocol. We 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 AIMD flow shares the link bandwidth with other cross traffic. In the following, we will introduce the QoS indexes for delay-sensitive multimedia applications and the characteristics of wireless links.

A. QoS Indexes

For delay-sensitive multimedia applications over best-effort and highly dynamic IP-based networks, scalable source coding schemes with error concealment property have been proposed and are anticipated to be widely deployed [2], [16], [17]. For instance, a multiple description (MD) coder partitions the source data into several sets and then compresses them independently to produce descriptions [17]. 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.

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, which measures the ratio of the number of packets failed to arrive at the multimedia receiver timely over the number of packets being sent by the sender. For delay-sensitive applications, packets suffering excessive delay become useless, and they will be discarded by the receiver. Besides packet losses in the network, delay outage events, which occur when end-to-end delay exceeds a prescribed threshold, are important components of packet losses for delay-sensitive applications. The delay outage probability should be bounded to avoid wasting network resources for useless packets. End-to-end delay has a deterministic part and a random part, and the latter is called delay jitter. Delay outage probability is equivalent to the probability of delay jitter exceeding a prescribed threshold D, $\Pr\{d_q > D\}$.

In summary, the QoS parameters are flow throughput, packet loss rate, and delay outage probability.

B. Wireless Link

Since wireless channels have inherent severer impairments, the transmission error rates over wireless links are non-negligible. For a given wireless channel and Forward Error Correction (FEC) coding scheme, the residual transmission error rate is still visible to the upper layer protocols, and the instantaneous error rate fluctuates from time to time.

To reduce the transmission errors visible to the upper layers, an effective error recovery scheme widely deployed in wireless links is the ARQ scheme: the link layer detects transmission errors and retransmits corrupted packets locally. For delay-sensitive applications, a low-persistent ARQ scheme is preferable [18]. Also, Channel-State-Dependent packet transmission and scheduling schemes are proposed to achieve multi-user diversity gain and to improve the link utilization, which also introduce throughput and delay variation over wireless links [19], [20].

With the link-level ARQ and CSD packet transmission schemes, the wireless link throughput, defined as the number of packets being successfully transmitted per unit time, is time-varying. The distribution of link throughput can be calculated given the wireless channel profile and the link layer protocol. For example, a widely accepted wireless channel model is the Rayleigh fading model, which can be represented by a finite-state Markov chain [21]. The Markov model can be built by discretizing the received instantaneous signal-to-noise ratio into several states. For a given modulation scheme, the channel profile based on the Markov model (*e.g.*, packet error rate for each state, state transition matrix, steady state probabilities) can be obtained [21]. In an extended technical report [22], we have derived the link throughput distributions for a Markov wireless channel with the link-level ARQ and a persistent or CSD packet transmission scheme.

IV. Parameter Selection

To efficiently support delay-sensitive applications in hybrid networks, the design objective of the transport layer protocol is to fairly share the network resources with coexisting TCP and AIMD flows, maximize the wireless link utilization and flow throughput, and provide satisfactory QoS.

A. TCP-friendly Condition

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 [23]. It has been shown in [7] 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: $\alpha = 3(1-\beta)/(1+\beta)$, where $0 < \alpha < 3$ and $0 < \beta < 1$. Different applications can choose one of the parameters, and the other one is calculated according to the TCP-friendly condition. For example, if the application can tolerate the sending rate being reduced by $1/8$, it can choose β equal to $7/8$, and the AIMD sender can set the parameter pair $(1/5, 1/8)$.

B. Wireless Link Utilization and Flow Throughput

With the AIMD congestion control mechanism, AIMD flows probe for available bandwidth and overshoot the network capacity frequently, which produces transient congestion and packet losses. If the bottleneck is a highly multiplexed link, dynamic probing is needed since the end-systems do not have the knowledge of the global traffic. However, for a cross-domain connection, the bottleneck is most likely the lightly multiplexed wireless link, and, in general, dedicated wireless links are allocated to multimedia flows. Although AIMD flows can efficiently utilize highly multiplexed links, the efficiency is not obvious for lightly multiplexed ones. In addition, the dynamics of available bandwidth in wireless links may not be due to the competition of coexisting flows, but due to the time-varying channel condition.

In the following, we first consider the scenario that one AIMD flow occupies a dedicated wireless link. Then, we study the multiple-flows scenario.

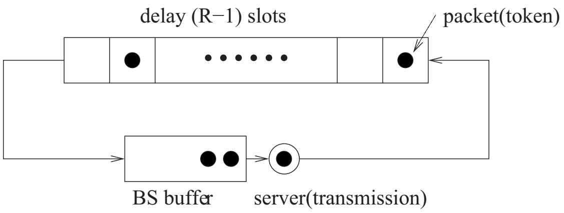


Fig. 4. Queue model.

1) Single AIMD Flow: As shown in Fig. 3, let a cross-domain AIMD flow occupy a time-varying wireless link with the link-level ARQ. We use the concept of *token* to emulate the window-based control: each packet (data or *ack*) needs a token to traverse the network, and the number of tokens (equal to W) is controlled by the sender according to the flow and congestion control mechanism. When the sender (or receiver) receives a packet accompanied by a token, it can transmit a packet with the token. Time is discretized into slots, each of which is long enough to transmit one packet over the wireless link.

Ideally, there is no delay jitter and packet losses in the wired domain, so the *rtt*, excluding queuing and retransmission delay at the BS, is a constant, denoted as R slots. As shown in Fig. 4, if a packet is transmitted over the wireless link successfully, the accompanying token is delayed for $R-1$ slots before it re-enters the BS buffer; if the transmission fails, the packet will be retransmitted in the next slot. The BS queue length (buffered tokens), Q, equals $W - X_{R-1}$, where X_{R-1} is the number of packets being successfully transmitted over the wireless link in $R-1$ slots. Obviously, $0 \leq X_{R-1} \leq R-1$.

When the wireless channel is in a bad condition and $X_{R-1} < W - B$ (*i.e.*, $Q > B$), packet losses due to buffer overflow will trigger the AIMD sender to exponentially reduce W. When channel condition becomes better, there may not be enough packets for transmission due to the small window size. Thus, the wireless link buffer size, B, can be conservatively set to *rwnd* to avoid BS buffer overflow (since $Q = W - X_{R-1} \leq rwnd - X_{R-1} = B - X_{R-1} \leq B$). This is achievable, *e.g.*, Enhanced GRPS system can set the buffer size up to 48 KB.

To maximize the link utilization, the wireless link should not be idle whenever the link layer decides to transmit, *i.e.*, Q should always be non-zero. Therefore, the sufficient condition to fully utilize the wireless link is $W \geq R$ (since $Q = W - X_{R-1} \geq R - X_{R-1} > 0$), which can be achieved if $rwnd = B \geq R$ (since W converges to $rwnd$ when $rwnd \leq R + B$). The maximum flow throughput is $1-p_e$ packet per slot, where p_e is the packet error rate of the wireless link.

However, in reality, delay jitter and packet losses may exist in the wired networks. Setting *rwnd* to R cannot absorb delay jitter in the wired domain. For instance, with $W = rwnd = R$, when all transmissions in the past $R-1$ slots are successful, the number of packets at the BS is only one; if a packet is delayed by one more slot in the wired domain, the BS queue will be empty for one slot. Such under-utilization of the wireless link can be avoided if $rwnd$ is larger than R. Also, when a packet loss occurs in the wired networks, the AIMD sender will exponentially reduce its sender window. If the reduced window size is less than R, the BS queue will be empty for some slots, and the wireless link is under-utilized. Therefore, it is desirable to set *rwnd* larger than R.

2) Multiple AIMD Flows: Let N AIMD flows share the wireless link. Without loss of generality, their deterministic *rtt*s satisfy $R_1 \leq R_2 \leq ... \leq R_N$. To maximize the wireless link utilization and flow throughput (*i.e.*, to guarantee $0 < Q \leq B$), we can set $B = S > R_N$, where S is the sum of all *rwnd*s, (because $0 < S - R_N + 1 \leq Q \leq S = B$).

C. Delay Outage Probability

With $B = rwnd$ or $B = S$, packet loss rate equals delay outage probability plus packet loss rate in the wired domain, and the latter can be estimated or bounded according to the historical measurements (*e.g.*, the data in [24]). Thus, bounding the packet loss rate is equivalent to bounding the delay outage probability. In the following, we focus on the delay outage probability.

1) Single AIMD Flow: Given $W = rwnd = B$, the probability of queuing delay exceeding D slots equals the probability of less than W successful transmissions in $(R+D)$ slots:

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

where the link throughput distribution $T_{R+D}(x)$ is the probability of x successful transmissions over the wireless link in $R+D$ slots.

From (1), delay outage probability is a non-decreasing function of W. The maximum W satisfying the desired delay outage probability is denoted as $W_{\max}$. By setting $rwnd = B \leq W_{\max}$, the delay outage probability can be bounded.

In summary, as shown in Section IV-B, with a larger *rwnd*, the BS queue can better absorb the delay jitter and packet losses in the wired domain to maximize the wireless link utilization and the flow throughput. On the other hand, to guarantee the delay outage probability, *rwnd* should be no larger than $W_{\max}$. Therefore, the optimal *rwnd* is $W_{\max}$.

2) Multiple AIMD Flows: Based on the analysis in [22], when N cross-domain AIMD flows share a wireless link, their window sizes and bandwidth ratios satisfy (2), where p_i and R_i are the i-th flow's bandwidth ratio and deterministic *rtt*, respectively, and $\mathrm{E}[T_{R_i-1}]$ is the mean number of successfully transmissions in $R_i - 1$ slots over the wireless link.

Denote $T_n(x;p_i)$ as the probability of successfully transmitting x packets of the i-th flow in n slots, *i.e.*, $T_n(x;p_i) = \sum_{k=0}^{n-x} \binom{x+k}{x} p_i^x (1-p_i)^k T_n(x+k)$. Obviously, $T_n(x;p_i+p_j) = T_n(x;p_i) + T_n(x;p_j)$ and $T_n(x) = T_n(x;1)$. Given $B = S = \sum_{i=1}^{N} rwnd_i \geq R_N$, the queue length distribution is shown in (3), where $\vec{W}$ represents the vector $\{W_1, W_2, \cdots W_N\}$ and $S - R_N + 1 \leq x \leq S$.

Given $B = S \geq R_N$, the delay outage probability is

$$\Pr\{d_q > D|\vec{W}\} = \sum_{x=S-R_N+1}^{S} \Pr\{Q = x|\vec{W}\} \sum_{j=0}^{x-1} T_{D+1}(j). \tag{4}$$

Under the constraint of (2), *rwnd*s can be set to the maximum integers satisfying delay outage probability calculated by (4).

$$W_i \;=\; p_i W_j / p_j + p_i(\mathrm{E}[T_{R_i-1}] - \mathrm{E}[T_{R_j-1}]), \qquad \text{for} \;\; 1 \le i, j \le N \tag{2}$$

$$\Pr\{Q = x|\vec{W}\} = \sum \cdots \sum_{y_1+y_2+\dots y_N = S-x} T_{R_1-1}(y_1;1) T_{R_2-R_1}(y_2; 1-p_1) \cdots T_{R_N-R_{N-1}}(y_N; p_N) \tag{3}$$

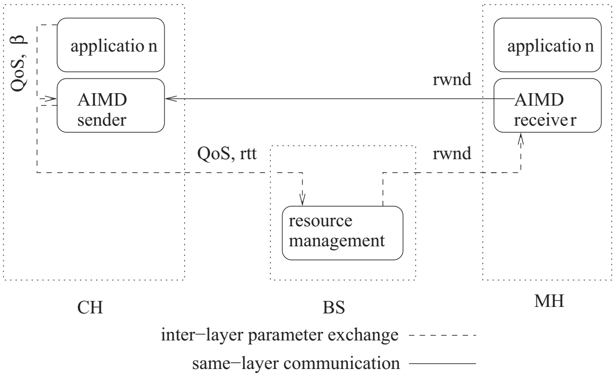


Fig. 5. Parameter exchange.

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 according to its maximum tolerable throughput variation.

(2) The application passes these QoS parameters to the transport layer protocol.

(3) In the transport layer, given β, the AIMD sender calculates α according to the TCP-friendly condition, and measures and estimates the *rtt*.

(4) The transport layer protocol passes the *rtt* and QoS parameters to BS.

(5) The BS resource allocation module (in the link layer) determines the wireless channel profile, chooses the link layer packet 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 $W_{\max}$ and B according to the QoS requirements and wireless link throughput distribution.

(7) The BS informs the AIMD receiver to set *rwnd* to the minimum of $W_{\max}$ and the actual available receiver buffer size. The AIMD receiver informs the AIMD sender the *rwnd*.

Parameter exchanges in steps (2), (4), (7) are shown in Fig. 5. 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 that the BS can check the table to determine appropriate *rwnd* and B. Therefore, the parameter selection procedure can be completed efficiently during 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, $W_{\max}$ is insensitive to small changes of the channel profile).

To efficiently support multimedia applications in hybrid wireless/wired networks, inter-layer interactions are necessary. Nevertheless, such interactions should be minimized for scalable system design purpose. As indicated in the above procedure, the parameter selection procedure requires only the 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 layer 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 wireless links with different channel models and physical/link layer protocols.

V. Simulation Results

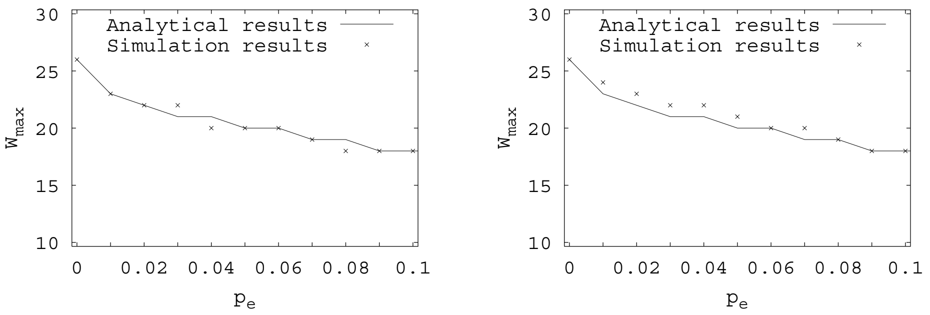
To verify the feasibility of our approach, examine link utilization, and evaluate protocol performance, we have performed extensive simulations with NS-2 [11].

The simulation topology is the same as that shown in Fig. 3. 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 connections share the $r_1 r_2$ 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 the BS and the MH are 200 Kbps and 100 Kbps, respectively. The buffer size at the BS is set to *rwnd* to avoid buffer overflow. The deterministic end-to-end delay for the target flow is 45 ms. 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.

The channel condition of the wireless link is dynamically changed according to a two-state Markov model [21]. In the good state, the transmissions over the wireless link are always successful, and in the bad state, the transmissions always fail. Thus, the packet error rate of the wireless link, p_e, equals the steady state probability of the bad state. The state transition probability from one state to the other state equals the steady state probability of the latter state. The link layer uses the persistent transmission scheme, *i.e.*, transmitting one packet each time slot no matter what the previous channel condition was.

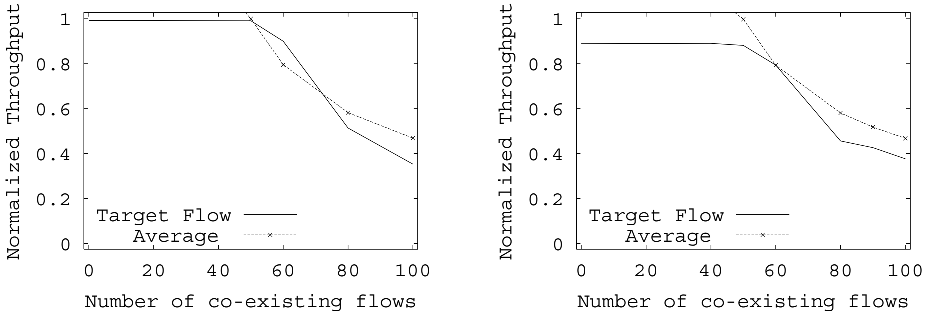
A. Delay Outage Probability

We repeat simulations with different values of *rwnd* and record the maximum *rwnd* with which less than 1% of



(a) light cross traffic (b) heavy cross traffic

Fig. 6. Maximum *rwnd* to guarantee delay outage probability.



(a) $p_e = 0.01$ (b) $p_e = 0.1$

Fig. 7. Normalized flow throughput.

the received packets with end-to-end delay (jitter) exceeding 95 ms (50 ms). The packet error rates of the wireless link, p_e, are set from 0% to 10%.

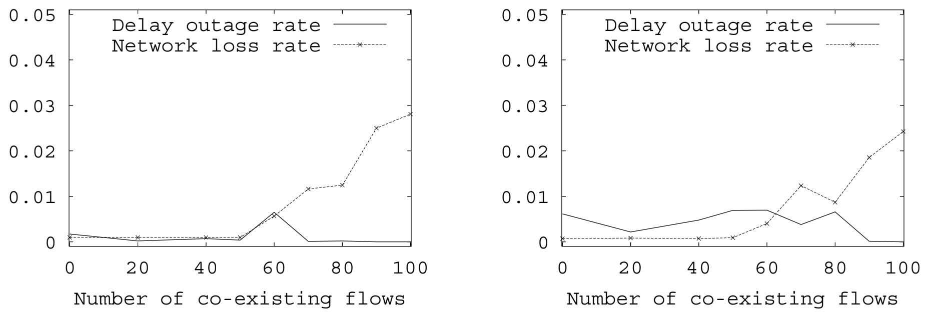
Simulations with light cross traffic and heavy cross traffic have been performed separately. In the light cross traffic case, the cross TCP traffic (both long-lived elephants and short-lived mice) in the backbone link ($r_1 r_2$) is constrained, so that the bottleneck for the target AIMD flow is always the wireless link; in the heavy cross traffic case, the coexisting TCP traffic volume is time-varying such that the bottleneck for the target AIMD flow is the backbone link between 30 and 60 seconds.

Figs. 6 shows the maximum window size $W_{\max}$ which can guarantee the delay outage probability below 0.01, w.r.t. p_e. It can be seen that the analytical results match well with the simulation ones when the cross traffic is light. As anticipated, the analytical results are slightly more conservative with heavy cross traffic, since when the bottleneck is the backbone link, $W = cwnd < rwnd = W_{\max}$. Simulations with other link layer transmission schemes (not shown here due to space limitation) have been performed (*e.g.*, suspending transmission for some slots if the previous transmission failed [19]), and the simulation results also closely approximate the analytical ones. In summary, no matter whether or not the bottleneck is the wireless link, and what transmission scheme is used in the link layer, it is feasible to calculate a suitable *rwnd* beforehand to bound the delay outage probability. Furthermore, Fig. 6 shows that $W_{\max}$ changes slowly w.r.t. p_e. For instance, when p_e is changed from 5% to 8%, $W_{\max}$ changes only by 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 *rwnd* (repeating steps (5) to (7) of the parameter selection procedure), and our scheme can tolerate a certain degree of errors of the wireless channel profile.

B. Wireless Link Utilization and TCP-friendliness

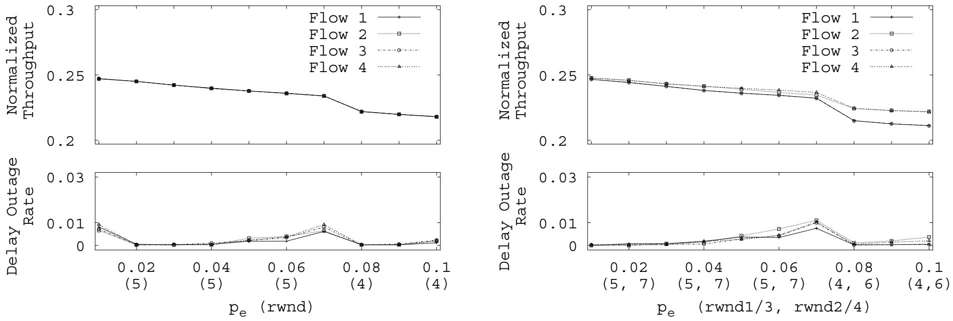
To examine link utilization and TCP-friendliness, the number of coexisting TCP flows (elephants) in the backbone link is changed from 0 to 100. The packet size of all TCP flows is 1250 bytes. Normalized flow throughput is defined as the number of packets being received (in the transport layer) per time slot. Figs. 7(a) and (b) show the normalized flow throughput of the target AIMD flow and the average normalized throughputs of all coexisting TCP and AIMD flows in the wired link, 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 one packet per slot, so the bottleneck for the target AIMD flow is the wireless link. To fully utilize the wireless link, the normalized throughput of the target flow should be $(1-p_e)$ packet per slot. When the number of coexisting flows is larger than 50, the bottleneck is the backbone link, and the normalized throughput of the target flow should be close to the average throughput in the backbone link for fairness. Fig. 7 shows that the AIMD flow achieves the desired performance. Simulation results with other value of p_e indicate the same tendency.

As shown in Fig. 8, the delay outage rate (ratio of the number of packets with delay exceeding the threshold over the number of packets being received) is less than 0.01.



(a) $p_e = 0.01$ (b) $p_e = 0.1$

Fig. 8. Delay outage rate and network loss rate.



(a) $R_1 = R_2 = R_3 = R_4 = 17$ (b) $R_1 = R_3 = 17$, $R_2 = R_4 = 25$

Fig. 9. Four AIMD flows sharing the wireless link.

The network loss rates [5] are negligible when the number of coexisting flows is less than 50, and the network loss rates increase to 2.8% when the number of coexisting flows is increased to 100, which indicates that the network packet losses are mainly due to network congestion in the wired domain. Therefore, with appropriate buffer size and window size, packet loss rate over the wireless domain and delay outage probability can be bounded.

C. Multiple AIMD Flows

The simulation topology for multiple AIMD flows sharing a wireless link is similar to that shown in Fig. 3, except that N AIMD senders are at N CHs which connect with r_1.

Let four AIMD flows have the same share of the wireless bandwidth. Their sender windows should satisfy (2), with $p_i = 0.25$. Also, to bound the delay outage probability, their sender windows are under the constraint of $\Pr\{d_q > D|\vec{W}\} \leq 0.01$. The *rwnd*s are set as the maximum integers satisfying the two constraints. The link buffer size is set to the sum of four *rwnd*s to avoid buffer overflow at the BS.

Fig. 9(a) shows the normalized throughputs and delay outage probabilities of four flows with the same deterministic *rtt*s of 17 slots, w.r.t. p_e. In Fig. 9(b), four AIMD flows have different deterministic *rtt*s: $R_1 = R_3 = 17$ slots and $R_2 = R_4 = 25$ slots. 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 deterministic *rtt*s or not.

Since *rwnd* is an integer, we may not be able to get a group of *rwnd*s to satisfy (2) exactly. Therefore, the results may slightly deviate from our designed target due to quantization errors, *e.g.,* in some case the throughputs of coexisting AIMD flows have small difference; in other case the delay outage probabilities exceed the desired 1% slightly. Nevertheless, such deviations can be anticipated and controlled.

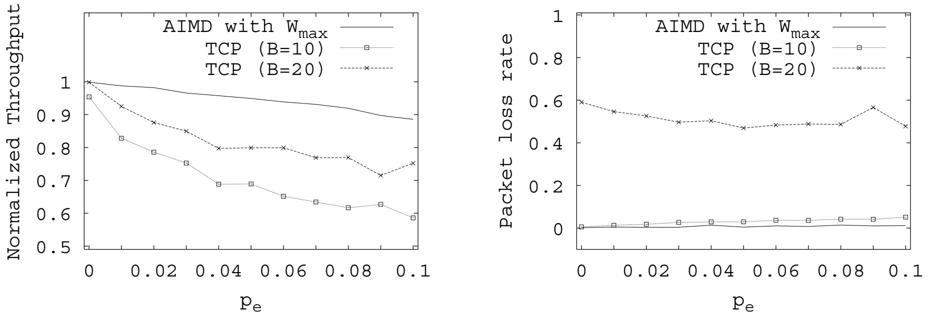
D. AIMD vs. TCP

We further evaluate the performance of AIMD flows with *rwnd* equal to $W_{\max}$ and TCP flows with *rwnd* equal to 50 packets. Fig. 10 compares the throughputs and packet loss rates (due to both buffer overflow and excessive delay) of the TCP and AIMD flows, w.r.t. p_e. For the TCP flows, we set the BS buffer size to 10 and 20, respectively. Fig. 10(a) shows that if the buffer size is larger, the throughput of TCP flow can be higher; but with a larger buffer, the delay outage probability increases significantly, as shown in Fig. 10(b). In addition, no matter which buffer size is chosen, the throughputs of TCP flows are less than that of AIMD flows, and the packet loss rates of TCP flows are higher than that of AIMD flows. This figure further demonstrates the advantages of the proposed protocol and its parameter selection scheme.

E. AIMD vs. UDP

The Internet has evolved from a small, research-oriented, and cooperative system to an enormous, commercial, and competitive information transport infrastructure. From a selfish users' point of view, they would like to discard any congestion

[5] The network loss rate measures the ratio of the number of packets being discarded in the network over the number of packets being sent by the sender.



(a) Normalized throughput (b) Packet loss rate

Fig. 10. AIMD vs. TCP.

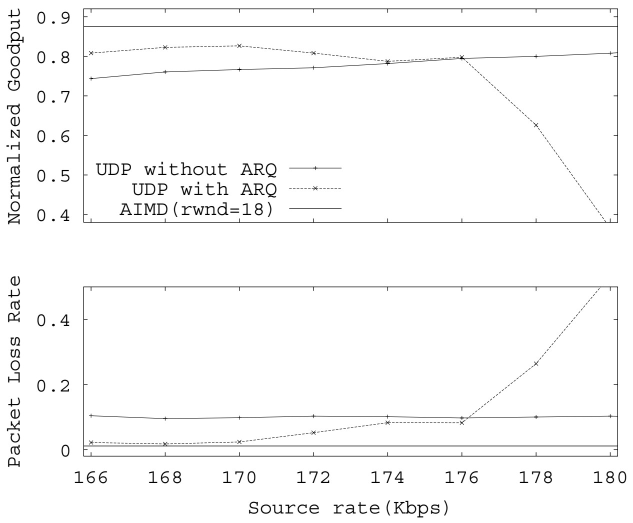


Fig. 11. AIMD vs. UDP, $p_e = 0.1$

control in their systems if such control has negative effects on their perceived QoS. How to punish the greedy or 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 error-prone wireless networks. This can be an incentive for the end-systems to deploy the AIMD congestion control.

Since the UDP protocol has no closed-loop control mechanism, the sender keeps sending packets regardless of network condition. Assume that the UDP sender has the knowledge of the wireless link, *i.e.,* link bandwidth, p_e, *etc.*, and it can choose its sending rate accordingly. Since the maximum tolerable delay jitter is 10 slots, the 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 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 within the delay bound per slot, which is equivalent to normalized throughput minus delay outage probability.

Fig. 11 compares the normalized goodput and packet loss rate for UDP (with and without ARQ) and AIMD-controlled flows, with $p_e = 0.1$. It is shown 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. With ARQ, the source rate of the UDP flow should be less than the average wireless link throughput to avoid excessive delay outage and packet loss rates. This is because, without window-based (self-clocking) control, UDP sender cannot adapt the sending rate when the BS queue is built up. From Fig. 11, no matter how the UDP sender adjusts its sending rate, and whether or not ARQ is deployed, 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 other words, the window-based AIMD protocol can provide better QoS since it can achieve higher goodput and lower packet loss rate.

VI. Related Work

Congestion control was incorporated into TCP protocol in the late 1980's when a series of congestion collapses were observed even when the Internet was relatively small [25]. The mainstream TCP congestion control variants, *i.e.*, TCP Tahoe, TCP Reno, TCP New Reno, TCP Sack, are all based on the AIMD congestion control mechanism, which uses packet losses as congestion signals with the assumption that they are mainly due to network congestion. However, this assumption may not hold in the wireless domain which has a noticeable transmission error rate. Efforts have been taken in the link layer to reduce packet losses due to transmission errors [26], [27]. FEC coding is used to enhance the error correction ability by introducing more redundancy. However, since wireless channels usually introduce burst errors, 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 scheme introduces more delay and delay jitter, which should be taken into consideration when designing transport layer protocol for delay-sensitive applications. The work reported in this paper serves this purpose.

Since TCP is not suitable for multimedia applications, two paradigms of TCP-friendly congestion control mechanisms have been proposed [2]–[7]: 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 probability is quite large, since the rate control mechanism does not have the acknowledgment self-clocking property, and it responds to channel dynamics more slowly than window-based control. Delay performance of TFRC flows over wireless links has been studied in [28].

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 [7]. Therefore, we choose window-based AIMD 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 [13], [29], [30]. In [13], *rwnd* is used to enhance fairness and reduce packet losses in 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 probability. In [29] and [30], *rwnd* is used to enhance TCP performance for hybrid ATM/IP and third generation wireless/wired networks, by adaptively adjusting the *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 [30] degrades when the *rtt* is above 100ms or the packet loss rate due to congestion in the wired domain exceeds 0.1%. In this paper, instead of frequently changing the *rwnd* according to the current queue length, we derive the optimal *rwnd* with consideration of the channel profile, flow *rtt*, and application QoS requirements, and the simulation results show the robustness of our approach.

Although we have listed extensive research works that are closely related to the transport layer protocol design for multimedia applications over wireless links, there are still a lot of challenge and open issues in this active area.

VII. Conclusions

A TCP-friendly AIMD protocol is proposed to support delay-sensitive multimedia applications over hybrid wireless/wired networks. By appropriately selecting the AIMD protocol parameters, wireless resources can be efficiently utilized, flow throughput can be maximized under the constraint of the delay outage probability. Simulation results have validated our analysis, demonstrated the feasibility of our approach, and shown that the AIMD protocol can outperform the non-responsive UDP protocol when they are used to support multimedia applications over hybrid networks. The presented research has focused on one-hop wireless infrastructure networks, extension to multi-hop wireless ad hoc networks is currently under investigation.

Acknowledgment

This work has been supported in part by a Postgraduate Scholarship and a Strategic Research Grant from the Natural Science and Engineering Research Council of Canada. We also want to thank Dr. Wanjiun Liao and the anonymous reviewers for their helpful comments on an early version of this paper.

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 Trans. 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] 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/. A shorter version appeared in Proceedings of ICNP'00, Osaka, Japan, November 2000.

[7] L. Cai, X. Shen, J. Pan, and J. W. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 339–355, Apr. 2005.

[8] 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.

[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] A. Evlogimenos, K. H. Lim, and K. Lai, "On the implementation of Datagram Congestion Control Protocol," 2002, available http://www.cs.berkeley.edu/~laik/projects/dccp/index.html.

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

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

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

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

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

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

[19] 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.

[20] P. Bhagwat, P. P. Bhattacharya, A. Krishna, and S. K. Tripathi, "Enhancing throughput over wireless LANs using channel state dependent packet scheduling," in *Proc. IEEEINFOCOM'96*, 1996, pp. 1133–1140.

[21] H. S. Wang and N. Moayeri, "Finite-state Markov channel-a useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, 1995.

[22] L. Cai, X. Shen, J. W. Mark, and J. Pan, "Performance modeling and analysis of window-controlled multimedia flows in wireless/wired networks," May 2004, technical report, Department of Electrical and Computer Engineering, University of Waterloo, Canada, available at http://bbcr.uwaterloo.ca/˜cai/tech-04-1.pdf.

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

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

[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] 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.

[28] 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*, vol. 3, 2004, pp. 1681–1685.

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

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

Lin Cai ((S'00-M'06) received the MASc and PhD degrees (with Outstanding Achievement in Graduate Studies Award) in electrical and computer engineering from the University of Waterloo, Waterloo, Canada, in 2002 and 2005, respectively. Since July 2005 she has been an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Victoria, Victoria, Canada. Her research interests span several areas in wireless communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic over wireless, mobile, and ad hoc and sensor networks. She serves as the Associate Editor for *EURASIP Journal on Wireless Communications and Networking (JWCN)* and the *International Journal of Sensor Networks (IJSNet)*.

Xuemin (Sherman) Shen (M'97-SM'02) received the B.Sc. (1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. From September 1990 to September 1993, he was first with the Howard University, Washington D.C., and then the University of Alberta, Edmonton (Canada). Since October 1993, he has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where he is a Professor. Dr. Shen's research focuses on mobility and resource management in interconnected wireless/wireline networks, UWB wireless communications systems, wireless security, and ad hoc and sensor networks. He is a coauthor of two books, an editor of 10 journal Special issues, and has published more than 150 papers in wireless communications and networks, control and filtering.

Dr. Shen was the Technical Co-Chair for IEEE Globecom'03 Symposium on Next Generation Networks and Internet, and ISPAN'04. He serves as the Associate Editor for *IEEE Transactions on Wireless Communications*; *IEEE Transactions on Vehicular Technology*; *ACM Wireless Networks*; *Computer Networks*; *Dynamics of Continuous, Discrete and Impulsive - Series B: Applications and Algorithms*; *Wireless Communications and Mobile Computing* (Wiley); and *International Journal Computer and Applications*. He also serves as Guest Editor for *IEEE Journal on Selected Areas in Communications*, *IEEE Wireless Communications*, and *IEEE Communications Magazine*. Dr. Shen received the Premier's Research Excellence Award (PREA) from the Province of Ontario, Canada for demonstrated excellence of scientific and academic contributions in 2003, and the Distinguished Performance Award from the Faculty of Engineering, University of Waterloo, for outstanding contribution in teaching, scholarship and service in 2002. Dr. Shen is a senior member of the IEEE, and a registered Professional Engineer of Ontario, Canada.

Jon W. Mark (M'62-SM'80-F'88-LF'03) received the B.A.Sc. degree from the University of Toronto in 1962, and the M.Eng. and Ph.D. degrees from McMaster University in 1968 and 1970, respectively, all in electrical engineering.

From 1962 to 1970, he was an engineer and then a senior engineer at Canadian Westinghouse Co. Ltd., Hamilton, Ontario, Canada. In September 1970 he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, where he is currently a Distinguished Professor Emeritus. He served as the Department Chairman during the period July 1984-June 1990. In 1996 he established the Centre for Wireless Communications (CWC) at the University of Waterloo and is currently serving as its founding Director.

Dr. Mark had been on sabbatical leave at the following places: IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a Visiting Research Scientist (1976-77); AT&T Bell Laboratories, Murray Hill, NJ, as a Resident Consultant (1982-83): Laboratoire MASI, Universite Pierre et Marie Curie, Paris France, as an Invited Professor (1990-91); and Department of Electrical Engineering, National University of Singapore, as a Visiting Professor (1994-95).

He has previously worked in the areas of adaptive equalization, image and video coding, spread spectrum communications, computer communication networks, ATM switch design and traffic management. His current research interests are in broadband wireless communications, resource and mobility management, and cross domain interworking. He recently co-authored the text entitled Wireless Communications and Networking, Prentice-Hall 2003.

Dr. Mark is a Life Fellow of IEEE. He is the recipient of the 2000 Canadian Award for Telecommunications Research and the 2000 Award of Merit of the Education Foundation of the Federation of Chinese Canadian Professionals, an editor of *IEEE Transactions on Communications* (1983-1990), a member of the Inter-Society Steering Committee of the *IEEE/ACM Transactions on Networking* (1992-2003), a member of the IEEE Communications Society Awards Committee (1995-1998), an editor of *Wireless Networks* (1993-2004), and an associate editor of *Telecommunication Systems* (1994-2004).

Jianping Pan (S'96-M'99) is currently an Assistant Professor of computer science at the University of Victoria, British Columbia, Canada. He received his BS and PhD degrees in computer science from Southeast University, Nanjing, China in 1994 and 1998, respectively. From 1999 to 2001, he was a postdoctoral fellow and then a research associate at the University of Waterloo, Ontario, Canada; from 2001 to 2005, he was a member of research staff at Fujitsu Labs and a research scientist at NTT MCL in Silicon Valley, California, USA. His area of specialization is distributed systems and networks, and his recent research interests include protocols for advanced networking, performance analysis of networked systems, and applied network security. He is a member of the ACM and the IEEE.