Performance Analysis of TCP-Friendly AIMD Algorithms for Multimedia Applications

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

***Abstract*—In this paper, the performance of TCP-friendly generic AIMD (Additive Increase and Multiplicative Decrease) algorithms for Web-based playback and multirate multimedia applications is investigated. The necessary and sufficient TCP-friendly condition is derived, and the effectiveness and responsiveness of AIMD are studied. Due to practical implications, a Dynamic TCP-friendly AIMD (DTAIMD) algorithm is proposed. Extensive simulation results are given to verify the derived necessary and sufficient condition, and to demonstrate the performance of the proposed DTAIMD algorithm.**

***Index Terms*—Congestion control, Internet, multimedia, AIMD, quality-of-service, TCP-friendly.**

I. Introduction

With THE universal adoption of the Internet as a global information infrastructure, besides the traditional Web applications, more and more highly-demanded and media-rich applications emerge and become very popular. Although the HTTP-based Web traffic still dominates the Internet, it is well anticipated that these new applications will claim a large percentage in the future traffic mix. The success of the Internet essentially relies on the self-regulated Transmission Control Protocol (TCP) in its transport layer. TCP does not assume any explicit knowledge about network internals and other sessions. If a congestion signal is captured, TCP sender aggressively reduces its *congestion window* (cwnd) by a half, or even reinitializes cwnd for severe congestion. Otherwise, TCP probes for unused bandwidth conservatively by enlarging cwnd by one segment per *round-trip time* (rtt).

Although it has been shown that TCP is very successful for bulk data transfer, there are several design issues that make it less attractive for multimedia applications: a) TCP couples flow, error, and congestion controls *together* to offer a single reliable data transfer service, while many multimedia applications have much tighter and more diverse requirements on delivery timeliness rather than object integrity; b) TCPs *increase by one or decrease by half* strategy produces a highly fluctuating sending rate which is undesirable for most multimedia applications; and c) TCP does not have some of the desirable features such as multipoint communications and group management. Therefore, the well-behaved TCP is rarely chosen to transport multimedia traffic over the Internet. Since TCP-transported applications are anticipated to be the dominant in the near future, it is crucial to have compatible traffic regulations for non-TCP applications. These regulations, or congestion control focused in this paper, are expected to achieve the following objectives: a) different classes of multimedia applications can coexist and behave properly and b) these multimedia applications can share network resource appropriately with ordinary TCP-transported applications. We refer to these regulations as *TCP-friendly congestion control for non-TCP-transported multimedia applications* in this paper. In addition to the fairness and TCP-friendliness issues, any new congestion control scheme should also i) have the ability to maintain network stability by promptly responding to congestion and be cooperative with other flows; ii) utilize network resources (e.g., link bandwidth) efficiently; and iii) be capable to provide better QoS, (e.g., a smoothed flow and bounded latency for playback multimedia applications); iv) be simple to implement, compatible with the legacy, and scalable for incremental deployment.

There are two paradigms of TCP-friendly congestion control schemes proposed in the literature: *equation-based rate control* and *generic Additive Increase Multiplicative Decrease* (AIMD) *based window control*. For the equation-based approach, several analytical models [9], [10] are used to obtain the long-term TCP throughput as a function of the measured packet[1] loss 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 sender does under the same condition. TCP-Friendly Rate Control (TFRC) is a representative scheme in this paradigm [5]. However, it is found that the throughput model is quite sensitive to parameters (e.g., p and rtt) that are difficult to measure efficiently and to predict accurately. Also, the long-term TCP throughput equation does not capture the transit and short-lived TCP behaviors, and it is less responsive to short-term network and session dynamics. Furthermore, it is unknown whether the model itself still applies when the mix

Manuscript received April 5, 2002; revised August 5, 2003. This work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) under Grant No: RGPIN7779 and a Postgraduate Scholarship, and grants from the Canadian Institute for Telecommunications Research (CITR) under the NCE program of the Government of Canada. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nelson L. S. Fonseca.

L. Cai, 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: cai@bbcr.uwaterloo.ca; xshen@bbcr.uwaterloo.ca; jwmark@bbcr.uwaterloo.ca).

J. Pan was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. He is now with NTT Multimedia Communications Laboratories, Palo Alto, CA 94086 USA (e-mail: jpan@nttmcl.com).

Digital Object Identifier 10.1109/TMM.2005.843360

[1]The terms network *packet*s and transport *segment*s are used interchangeably in this paper since the modern TCP can negotiate for the maximum segment size on its connection establishment to avoid IP fragmentation.

of TCP and non-TCP traffic changes dramatically over time as new applications keep emerging over the Internet. On the other hand, the Generic AIMD (GAIMD) based approach inherits the same principle embedded in TCP. Instead of the *increase by one or decrease by half* strategy, GAIMD increases its cwnd by α segments additively when no congestion is sensed; otherwise, it multiplicatively decreases the cwnd to β of its previous value. TCP is a special case of GAIMD with $(\alpha = 1, \beta = 0.5)$. With an appropriate pair of (α, β), it is possible to have a smoothed sending rate and at the same time to be TCP-friendly. TCP-Friendly GAIMD (TFGAIMD) [6] is such a scheme in this paradigm. It derives the α and β pair from the AIMD response function, which is an extension of the TCP response function. However, due to this indirect approach, the only TFGAIMD pair with the TCP-friendliness and independence of p is $\alpha = 1$ and $\beta = 0.5$, i.e., TCP itself. There are other variants proposed in the literature, e.g., RAP [3] explores the inter-packet gap (IPG) to exercise congestion control. If no congestion is sensed, IPG is reduced additively; otherwise, IPG is doubled multiplicatively. This technique can also be applied to inter-acknowledgment spacing based approaches [4].

In this paper, instead of proposing another scheme either in the window or rate based control paradigm, we focus on the generic AIMD algorithms and their α and β parameters, in particular for multimedia applications over the Internet. Since GAIMD inherits many TCP control properties, it is more feasible to implement and deploy GAIMD in a large scale for non-TCP applications. Despite some previous work on GAIMD protocols, the competitive behaviors among TCP and GAIMD, especially when GAIMD becomes the dominant component, are still far from being well understood. Therefore, we pursue a deeper understanding of the intrinsic properties and performance of GAIMD. Our main contributions are as follows. Without the constraints of any TCP long-term throughput equations or response functions, we introduce a novel approach to obtain the *necessary* and *sufficient* condition for a family of AIMD parameters to be TCP-friendly. Furthermore, we demonstrate that the condition is not only valid for any two TCP and AIMD flows but also captures the competitive behaviors among any number of multiclass AIMD flows. Based on the analysis of its built-in effectiveness and responsiveness properties and the discussion of practical implications, we propose a Dynamic TCP-friendly AIMD (DTAIMD) algorithm for Web-based playback and multirate multimedia applications. The efficacy and performance of the TCP-friendliness condition and the DTAIMD algorithm are analyzed along with intensive performance studies.

The rest of this paper is organized as follows. In Section II, we briefly overview the TCP and TCP-friendly congestion control schemes. In Section III, we analyze the AIMD algorithms with a focus on their (α, β) pairs. The models for one TCP and one AIMD, and for multiple-class AIMD flows, are established to obtain the necessary and sufficient TCP-friendly condition. The effectiveness and responsiveness properties of such condition and its practical implication are also discussed with the DTAIMD proposal. The DTAIMD-based congestion control for multimedia playback and multirate applications, including service differentiation and low or high multiplexing scenarios, are then studied in detail in Section IV. Performance studies are presented in Section V, which includes the simulation results for the flow fairness, link utilization, convergence speed, service differentiation and QoS provisioning of DTAIMD-based multimedia applications. Section VI gives concluding remarks and issues for future work.

II. Related Work

In this section, we first outline the mainstream TCP congestion control mechanisms and Active Queue Management (AQM) capable routers. We then briefly enumerate the existing TCP-friendly congestion control schemes.

A. TCP Congestion Control

Congestion control was incorporated into TCP flow and error controls in the late 1980s when a series of congestion collapses were observed even when the Internet was relatively small at that time [1]. The main principle of TCP congestion control is *packet conservation* (also known as *acknowledgment self-clocking*): a new segment is injected into network only after an old one has left. This is an ideal case in the network and session equilibrium state. However, changes in available network resources and number of sessions always deviate TCP away from the ideal case. Therefore, TCP congestion control is necessary to maintain network stability [2], and the Internet evolution in the last two decades has proved this. For TCP congestion control, initially, after a timeout, or being idle for a while, TCP probes for available network resource with a small cwnd. It is known as **Slow Start**. In order to reach the target equilibrium quickly, TCP increases cwnd exponentially every rtt, until cwnd is above the *slow start threshold* (ssthresh) or when congestion occurs. When $\mathrm{cwnd} > \mathrm{ssthresh}$, TCP probes for the unused bandwidth conservatively, i.e., increasing cwnd by one segment per rtt without delayed acknowledgment. This is referred to as **Congestion Avoidance**, until eventually congestion occurs. For the mainstream TCP variants, timeout and duplicated acknowledgment (dupack) are two common congestion indications. Timeout indicates severe congestion, and forces TCP to reinitialize cwnd and halve ssthresh. Normally, the occurrence of three dupacks are considered to signify moderate congestion, so cwnd is halved (or more precisely, reduced to a half amount of current outstanding data) and $\mathrm{ssthresh} = \mathrm{cwnd}$. Therefore, TCP is characterized as AIMD(1, 0.5).

With the wide adoption of TCP/IP protocols in the Internet, TCP congestion control also continues its evolution. New algorithms, e.g., Partial Acknowledgment [12] and Selective Acknowledgment [13], have been proposed and incorporated into TCP endpoints. For intermediate systems, traditionally **Drop-Tail** routers drop incoming packets whenever their output buffer overflows, as a congestion signal to notify responsive transport endpoints. Drop-Tail queuing is known to produce bursty packet losses and a bias against flows with long rtts and small packets. Currently, modern AQM-capable routers can detect congestion even before buffer overflow actually occurs. Random Early Detection (RED) is a well-known AQM scheme [11]. RED-capable routers monitor the queue length (and the speed at which it increases). If the length is below a

lower threshold, no packet is dropped; if it is above an upper threshold, all packets are dropped. When the queue length is between these two thresholds, packets are dropped with a certain probability. With RED, it is very likely that the packet loss probability of a flow is proportional to its packet rate, and the congestion signals can be distributed among different flows more fairly. Furthermore, Explicit Congestion Notification (ECN) [14] is proposed in both the network and transport layers. ECN-capable RED routers can advertise the incoming congestion without dropping packets, and ECN-capable endpoints can exercise congestion control proactively. There are other queuing mechanisms such as Fair Queuing (FQ) and Weighted FQ (WFQ) in the literature that offer better fairness with higher complexity [15].

B. TCP-Friendly Congestion Control

TCP is rarely chosen for emerging multimedia applications, but the traffic regulation is necessary for these applications. Therefore, TCP-friendly congestion control has become an active research topic recently [3]–[7]. Since the instantaneous TCP sending rate is highly complicated due to network and session dynamics, **TCP-friendliness** is defined as the average throughput of non-TCP-transported applications over a large time scale does not exceed that of any conformant TCP-transported ones under the same circumstance [8]. We use this definition in our work.

Many studies have been done on the quantitative modeling of the TCP long-term throughput as a function of p, rtt, MSS, and T_0. A simple TCP throughput equation [9] which assumes that dupack is the only congestion signal during Congestion Avoidance is $C_0\mathrm{MSS}/(\mathrm{rtt}\sqrt{p})$, where $C_0 = \sqrt{3/2}$ without the delayed acknowledgment policy. A more sophisticated equation [10], referred to as the TCP response function, is

$$R = \frac{\mathrm{MSS}}{\mathrm{rtt}\sqrt{\frac{2p}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3p}{8}}\right) p(1+32p^2)} \quad (1)$$

which also takes the sender timeout into account.

TFRC [5] utilizes the TCP response function directly and regulates its sending rate by calculating sending rate R with the measured p, rtt, and the negotiated MSS. With this equation-based rate control, it is possible to have a smooth sending rate, given the assumption that the product of $\sqrt{p}$ and rtt will also remain smooth. TFRC is still a closed-loop control mechanism since it needs to measure the latest p and rtt periodically, although it has more flexibility on generating acknowledgment than the AIMD-based mechanisms. Since TFRC relies on the TCP response function, the accuracy of the measured p and rtt becomes crucial. In practice, these two parameters are very difficult to measure effectively and predict accurately. TFGAIMD [6] also utilizes this response function to derive the TCP-friendly (α, β) pairs in an indirect approach. With their approach, the only (α, β) pair being TCP-friendly and independent of p is just TCP itself, i.e., $\alpha = 1$ and $\beta = 0.5$. This pair only produces a highly fluctuating sending rate that is undesirable for most multimedia applications.

RAP [3] also adopts the generic AIMD approach by adjusting the IPG additively or multiplicatively, depending on whether or not congestion signal is captured. The window-based AIMD approaches can potentially introduce burstiness into flows when cwnd is advanced or inflated rapidly. As a rate-based control, RAP avoids this problem by spacing outgoing packets evenly to produce a smoothed flow. Our work focuses on the relation between α and β which can be translated into the additive and multiplicative parameters to adjust IPG. Therefore, our work is separate from RAP and can be used to enhance the RAP performance.

Besides AIMD-based congestion control, other schemes based on binomial congestion control were proposed in the literature [7], for instance, Inverse Increase and Additive Decrease (IIAD), Square-root Increase and Square-root Decrease (SQRT), etc. These schemes can increase or decrease cwnd more smoothly and are attractive for multimedia applications. However, although they are TCP-friendly under certain circumstances, it is impossible for them to be TCP-friendly independent of the bottleneck link capacity. We have further comparison on AIMD and the binomial-based schemes in Section III-A2.

Different from these previous approaches, we study the competitive behaviors of TCP and AIMD flows, and directly derive the necessary and sufficient condition for the AIMD increase and decrease parameter pairs to be TCP-friendly. This condition itself is independent of link capacity due to the inherited properties of AIMD algorithms. Our direct approach avoids the TCP response function as well as its assumptions and limitations. With the TCP-friendly condition, a family of parameter pairs can be chosen for different applications with respect to their own traffic characteristics. The derived parameter pairs are also applicable for other AIMD-based congestion control schemes, e.g., TCP and RAP. We demonstrate that better quality of service provisioning, in terms of smoother packet flow, less buffer underruns, and less delay jitter, is achievable with the dynamic TCP-friendly AIMD algorithm. Furthermore, based on the derived parameter family, we can effectively provide service differentiation to multiclass multimedia applications, without any additional changes in the core network.

III. Analysis on AIMD Algorithms

We first model the AIMD algorithms to derive their TCP-friendly condition in this section, and analyze the properties of AIMD on effectiveness and responsiveness. Practical implications are then discussed, and a dynamic TCP-friendly AIMD algorithm is proposed.

A. TCP-Friendly AIMD Parameters

Generic AIMD has the identical Slow Start process initially or after timeout as that of TCP. However, in Congestion Avoidance, instead of using TCPs *increase by one or decrease by half* strategy, GAIMD adjusts its cwnd according to two parameters: α and β. Therefore, we only need to examine the competitive behaviors between TCP and AIMD in Congestion Avoidance. For window-based schemes, an AIMD sender increases cwnd by α packets per rtt to probe for unused bandwidth when there

is no congestion, and decreases to β times its previous cwnd if a congestion signal is captured. For spacing-based schemes like RAP, the sender decreases IPG to $1/(1+\alpha)$ of its previous IPG if no congestion presents; otherwise, it increases IPG to $1/\beta$ of the previous value. For presentational simplicity, we only use the window-based one as an example in this paper, and our results are also applicable to the spacing-based schemes.

1) One TCP and One AIMD Flows: Let one TCP flow and one AIMD(α, β) flow share a link with capacity r packets per rtt (i.e., r packets can be transmitted during one rtt)[2]. Assume that both flows have the same rtt and MSS, and the effects due to different rtt and MSS will be discussed in Section III-A2. Fig. 1(a) shows their cwnd trajectory. The cwnds of the AIMD flow and TCP flow at time t are denoted as $W_A(t)$ and $W_T(t)$, respectively. When $W_T(t)+W_A(t) < r$, it is referred to as the *underload* region, and W_T and W_A evolve as follows

$$W_A(t+\Delta t) = W_A(t) + \alpha \cdot \Delta t \tag{2}$$

$$W_T(t+\Delta t) = W_T(t) + 1 \cdot \Delta t. \tag{3}$$

Combining (2) and (3), we have

$$\frac{W_A(t+\Delta t) - W_A(t)}{W_T(t+\Delta t) - W_T(t)} = \alpha. \tag{4}$$

Therefore, the cwnd trace in the *underload* region follows a line segment with a slope α.

If $W_T(t)+W_A(t) \geq r$, it is referred to as the *overload* region. Denote t_i the time when the trace enters the *overload* region for the i-th time, where $i \geq 0$. Assume that both flows receive the congestion signal once the sum of their sending rates overshoots the link capacity, and decrease their cwnds simultaneously, we have

$$W_A(t_i) + W_T(t_i) = r \tag{5}$$

$$W_A(t_i^+) = \beta W_A(t_i) \tag{6}$$

$$W_T(t_i^+) = 0.5 W_T(t_i). \tag{7}$$

As indicated in (6)–(7), the cwnd trace has a slope of $((1-\beta)W_A(t_i))/((1-0.5)W_T(t_i))$ when congestion occurs. Consequently, the cwnd trace is in the *underload* region again. The duration between t_i and t_{i+1} is referred to as the i-th *cycle*. The increase and decrease *cycle*s repeat as the TCP and AIMD keep probing for unused bandwidth.

Based on (2)–(7), we have

$$W_A(t_{i+1}) - W_A(t_i) = -\frac{2(1-\beta)+\alpha}{2(\alpha+1)} W_A(t_i) + \frac{\alpha \cdot r}{2(\alpha+1)}. \tag{8}$$

If $W_A(t_i) > (r\cdot\alpha)/(2(1-\beta)+\alpha)$, the cwnd decrease slope of the i-th *cycle* is greater than α, and $W_A(t_{i+1}) < WAt_i)$, i.e., $W_A(t_i)$ monotonically decreases and converges to $(r\cdot\alpha)/(2(1-\beta)+\alpha)$. If $W_A(t_i) < (r\cdot\alpha)/(2(1-\beta)+\alpha)$, the decrease slope in the i-th *cycle* is less than α, and $W_A(t_{i+1}) > W_A(t_i)$, i.e., $W_A(t_i)$ monotonically increases and again converges to $(r\cdot\alpha)/(2(1-\beta)+\alpha)$. Similarly, $W_T(t_i)$ converges to $(r\cdot 2(1-\beta))/(2(1-\beta)+\alpha)$. Therefore, independent of the initial values of $W_A(0)$ and $W_T(0)$, after a sufficient number of *cycle*s, the cwnd trace of these two flows in the *overload* region converges to

$$W_A(t_*) = \frac{r\cdot\alpha}{2(1-\beta)+\alpha} \tag{9}$$

$$W_T(t_*) = \frac{r\cdot 2(1-\beta)}{2(1-\beta)+\alpha} \tag{10}$$

where t_* represents the time of *overload* in steady state. From (9) and (10), in steady state, W_A and W_T increase and decrease periodically, oscillating along the line segment with the slope α, as shown in Fig. 1(a). Their average throughputs are proportional to their average cwnds in steady state, i.e.,

$$\overline{W_A} = \frac{(1+\beta)}{2} W_A(t_*) = \frac{(1+\beta)\alpha r}{4(1-\beta)+2\alpha} \tag{11}$$

$$\overline{W_T} = \frac{(1+0.5)}{2} W_T(t_*) = \frac{3(1-\beta) r}{4(1-\beta)+2\alpha}. \tag{12}$$

To guarantee that the TCP and AIMD flows have a fair share of the link capacity[3], the *necessary* and *sufficient* condition is $\overline{W_A} = \overline{W_T}$. From (11) and (12), we have

$$\alpha = \frac{3(1-\beta)}{1+\beta} \tag{13}$$

where $0 < \alpha < 3$ and $0 < \beta < 1$. The larger the β is, the smaller the α should be.

Using the same technique, the TCP-friendly condition can be extended for any two classes of AIMD flows. For AIMD(α_1, β_1) and AIMD(α_2, β_2), they are friendly if

$$\frac{\alpha_1}{\alpha_2} = \frac{(1+\beta_2)(1-\beta_1)}{(1-\beta_2)(1+\beta_1)} \tag{14}$$

where $\alpha_1, \alpha_2 > 0$ and $0 < \beta_1, \beta_2 < 1$.

Equations (13) and (14) are valid as long as the competing flows have the same chance to reduce their cwnds. With RED-

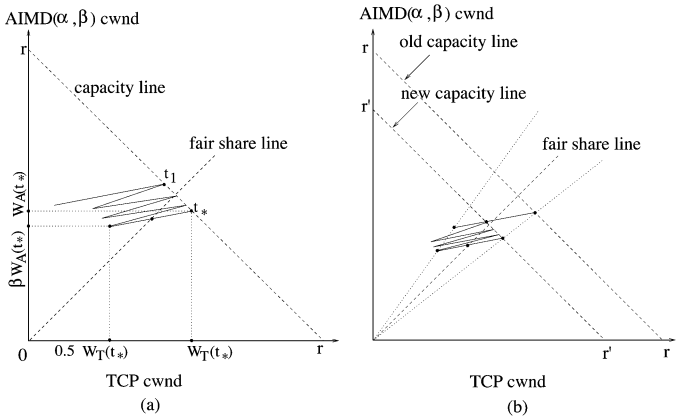

Fig. 1. The cwnd traces for one TCP flow and one AIMD flow: (a) static link capacity and (b) dynamic link capacity.

[2]In this paper, cwnd is calculated in the number of full size packets, and *time* is in the number of rtts.

[3]For presentational simplicity, we assume all flows have the same weight or priority in this section. When they have different weights as discussed in Section IV-C, the friendly conditions need to be updated with a scalar factor.

capable routers, packets are discarded or marked proportionally when the queue length exceeds a threshold. Therefore, it is reasonable to assume the competing flows receive a fair share of congestion signals, unless cwnd of one flow is much larger than the others. If so, that flow has a better chance to receive congestion signals and reduces its cwnd more frequently. Therefore, the cwnd trace converges to the steady state even faster.

a) Dynamic link capacity: Equation (13) indicates that the TCP-friendly condition is actually independent of the link capacity r. As shown in Fig. 1(b), when the link capacity changes from r to r', the cwnd trace converges to a new line segment with the slope α. One endpoint of the line segment is on the line $W_A = \alpha W_T/2(1-\beta)$, and the other endpoint is on the line $W_A = \alpha\beta W_T/(1-\beta)$. As far as (13) is satisfied, the midpoint of the new segment always sits on the fair share line, which means that the average throughputs of the AIMD and TCP flows are the same in steady state.

Yang *et al.* [6] also obtained the same equation as (13) by simplifying the AIMD response function. The simplification is based on the assumption that the packet loss rate (p) should be very small. Equation (13) was interpreted as a *partial* TCP-friendly condition for AIMD flows if triple-dupack is the only congestion signal. When timeouts are considered, they gave another condition as follows

$$\alpha = \frac{4}{3}(1-\beta^2). \tag{15}$$

The only (α,β) pair satisfies both (13) and (15) is just (1, 0.5), i.e., the pair for TCP only. However, after timeout, both TCP and AIMD have the same Slow Start process and thus the same throughput. We believe that it is sufficient to get the TCP-friendly condition by just focusing on Congestion Avoidance and exponential backoff due to triple-dupack. Reference [6] obtained (13) with the constraint and assumption of a small p. On the contrary, our straightforward approach derives (13) without any simplification and has no constraint on p. More discussions on the validation of the TCP-friendly condition (13) are presented in the following section, and the simulation studies in Section V-A1 confirm our analysis.

2) Multiclass AIMD Flows: Due to the heterogeneous nature, different multimedia applications can choose different (α,β) pairs, according to their unique traffic characteristics. Although the TCP-friendly condition is derived by studying the competitive behaviors of two flows, it will be demonstrated that multiclass AIMD flows can also co-exist friendly. Let n AIMD flows share a link with capacity r packets per rtt, the k-th AIMD flow have the pair (α_k,β_k), and its cwnd at time t be $W_k(t)$. (α,β) pairs should satisfy the following equation to be friendly pairwisely

$$\frac{\alpha_k(1+\beta_k)}{(1-\beta_k)} = \text{constant}, \quad k = 1,2,\ldots,n. \tag{16}$$

In the *underload* region, their cwnds evolve as

$$W_k(t+\Delta t) = W_k(t) + \alpha_k \cdot \Delta t. \tag{17}$$

In the *overload* region, assume that all flows get the congestion signal. We have

$$\sum_{k=1}^{n} W_k(t) = r \tag{18}$$

$$W_k(t^+) = \beta_k W_k(t). \tag{19}$$

Initially, the cwnd of the k-th flow is $W_k(0)$. Let t_i denote the time when the cwnd trace enters the *overload* region for the i-th time, and $\Delta t_i = t_{i+1} - t_i (i > 0)$. Δt_0 is the time from the beginning to the first time in the *overload* region. From (17)–(19), we have

$$W_k(t_i) = (W_k(0) + \alpha_k \Delta t_0)\beta_k^{i-1} + \sum_{j=1}^{i-1} \alpha_k \Delta t_j \beta_k^{i-1-j} \tag{20}$$

where $i > 0$. Since $\beta_k < 1, \beta_k^{i-1} \to 0$ when i is large. Therefore, the contribution of the initial value of cwnd fades out exponentially and becomes negligible after several cycles.

According to the analysis in Section III-A1, for two arbitrary flows, k and m, their cwnd trace in the *overload* region satisfies the following equation in steady state, no matter what the link capacity is

$$\frac{W_k(t_*)}{W_m(t_*)} = \frac{\alpha_k(1-\beta_m)}{\alpha_m(1-\beta_k)}, \quad \forall k,m. \tag{21}$$

By substituting (21) into (18), in steady state, the cwnd of the k-th flow in the *overload* region is

$$W_k(t_*) = \frac{r}{\sum_{j=1}^{n} \frac{\alpha_j}{1-\beta_j}} \cdot \frac{\alpha_k}{1-\beta_k}. \tag{22}$$

Thus, the average cwnd of the k-th flow in steady state is

$$\overline{W_k} = \frac{1+\beta_k}{2} W_k(t_*) = \frac{r}{\sum_{j=1}^{n} \frac{\alpha_j}{1-\beta_j}} \cdot \frac{\alpha_k(1+\beta_k)}{2(1-\beta_k)}. \tag{23}$$

Since all (α,β) pairs satisfy the friendly condition (16), $\overline{W_k}$ is a constant for all $k = 1,2,\ldots,n$. In conclusion, as long as their (α,β) pairs satisfy the friendly condition, in a long term, all AIMD flows eventually have an equal share of the link capacity, no matter how large the link capacity is, how many AIMD classes may exist, and how many flows are from each class.

b) Variable packet size and rtt: TCP is known for its bias against flows with small packets and long rtts, and the same is true for the TCP-friendly AIMD. The TCP-friendly AIMD flows have the same *packet* rate as the competing TCP flows. However, the throughput of an individual AIMD or TCP flow, in terms of bit per second, is proportional to its average packet size. One possible remedy is to give the small packet flows a higher weight to have a higher packet rate as discussed in Section IV-C. For different rtts, since the AIMD flow increases its cwnd by α packet per rtt, the effective increase rate is inversely proportional to its rtt. How to achieve fairness for small packet and long rtt

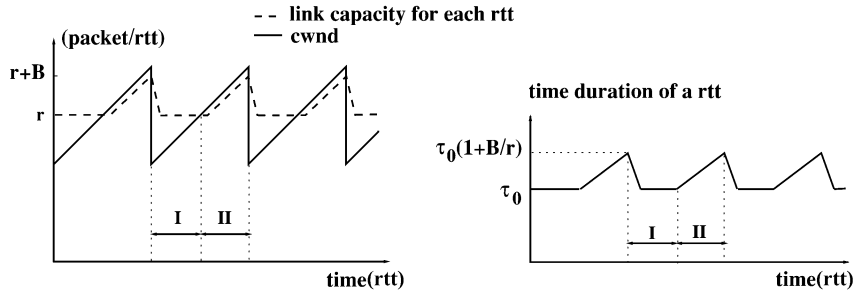

Fig. 2. The cwnd versus throughput for AIMD flows.

flows is beyond the scope of this paper and is one issue in our future work.

c) Comparison with binomial schemes: In steady state, the ratio of average cwnd of competing AIMD flows is independent of link capacity. With this important property, we can choose multiple (α, β) pairs according to the TCP-friendly condition. The flows in different classes with different (α, β) pairs can co-exist friendly. Although binomial increase/decrease schemes, e.g., IIAD and SQRT, etc., are attractive to multimedia applications for their smooth throughput, they cannot achieve the TCP-friendliness independent of link capacity, since in steady state, the ratio of bandwidth share of an AIMD flow and a co-existing non-AIMD flow depends on the bottleneck link capacity.

In the Appendix, we study the case when an IIAD flow competes with a TCP flow. No matter what parameters are chosen, the link capacity affect the bandwidth share of these two flows. Therefore, they cannot be friendly all the time. Another property of the IIAD scheme (derived in the Appendix) is: unlike AIMD, two IIAD flows can be friendly independent of link capacity if and only if these two IIAD flows have the same increase and decrease parameters.

B. Properties of AIMD Algorithms

Based on the analysis in Section III-A, the TCP-friendliness property is obvious for AIMD flows satisfying the TCP-friendly condition. Next, we explore two other important properties for AIMD flows: effectiveness and responsiveness.

1) AIMD Effectiveness: One effectiveness benchmark is network *utilization*, i.e., how effectively the AIMD-based protocols can utilize the available bandwidth when comparing with TCP.

a) Single AIMD flow: Let one $\mathrm{AIMD}(\alpha, \beta)$ flow occupy a link with fixed capacity. Without considering queuing delay, the rtt is constant and denoted as τ_0. The link can transmit r packets per τ_0, and the bottleneck buffer size is B packets. For RED-capable routers, B is defined as the mean value of queue length when packet loss occurs. As shown in Fig. 2, in steady state, there are two stages. In stage $\mathtt{I}, \mathrm{cwnd} < r$, there is no queuing delay and $\mathrm{rtt} = \tau_0$. In stage $\mathtt{II}, \mathrm{cwnd} > r$ and the queue builds up. For each rtt, cwnd increases by α packet and so does the queue. Therefore, rtt increases $\alpha\tau_0/r$ for every round[4]. The link utilization is 1 in stage II since the queue is nonempty all the time. When the cwnd increases to $r + B$, there are B packets in the queue, and packet loss occurs. Consequently, the cwnd is throttled to $\beta(r + B)$. In stage I, the number of packets being transmitted is $B+[r^2-\beta^2(r+B)^2]/2\alpha$. The time duration in stage I is $\tau_0[r - \beta(r + B)]/\alpha$. Therefore the link utilization in stage $\mathtt{I}, \eta_{\mathtt{I}}$, is

$$\eta_{\mathtt{I}} = \begin{cases} \frac{r^2-\beta^2(r+B)^2+2\alpha B}{2r^2-2\beta r(r+B)} & \text{when } 0 \le B \le B', \\ 1 & \text{otherwise} \end{cases} \tag{24}$$

where $B' = (\sqrt{(\alpha)/(2\beta^2) + ((1/\beta) - 1)r} - \sqrt{(\alpha)/(2\beta^2)})^2$.

Equation (24) indicates that TCP-friendly AIMD flows with a larger β only require a smaller buffer to fully utilize the link. On the other hand, although a large buffer can further improve the link utilization, it also introduces more delay and jitter, which is undesirable for delay sensitive multimedia applications. For these reasons, it is preferable to have a moderate network buffer size, especially for multimedia applications.

Without network buffering, the link utilization for an AIMD flow is

$$\eta = (1 + \beta)/2 \quad \text{where} \quad 0 < \beta < 1 \tag{25}$$

which indicates that the effective link utilization of an AIMD flow is proportional to $(1 + \beta)$.

b) Multiple AIMD flows: When there are n $\mathrm{AIMD}(\alpha, \beta)$ flows sharing a link, the link utilization becomes much more complicated to analyze since it also depends on how these flows synchronize in response to congestion signals.

The worst case scenario is that all flows simultaneously increase and decrease their windows. The link utilization in such case is exactly the same as when one $\mathrm{AIMD}(n\alpha, \beta)$ flow traverses the link. Without network buffering, the link utilization in the worst case, η_{worst}, is the same as (25).

The best case scenario is shown in Fig. 3, where the cwnd of each flow increase and decrease periodically, while the decreasing moments of each flow are evenly distributed. Neglecting network buffering, we have the following equation in the *overload* region

$$r = \sum_{i=1}^{n} (\beta W_{\max} + i(1 - \beta)W_{\max}/n)\,. \tag{26}$$

After a flow reduces its cwnd, the sum of all cwnds is $(r - (1 - \beta)W_{\max})$, and the sum will increase by $n\alpha$ packets per rtt in the

[4]A round is defined as the time successfully transmitting and receiving a whole cwnd of packets and acknowledgments.

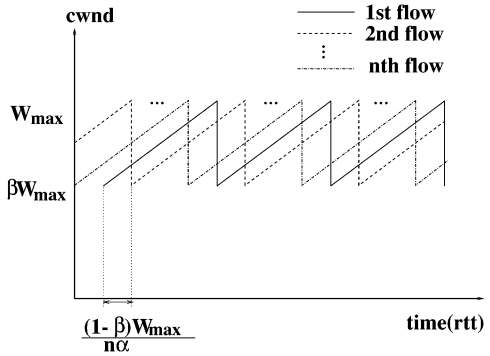

Fig. 3. n AIMD(α, β) flows in best scenario.

underload region. Therefore, the link utilization in the best case is

$$\eta_{\text{best}} = 1 - \left[\frac{(1+\beta)n}{1-\beta} + 1\right]^{-1}. \tag{27}$$

Equation (27) shows that AIMD flows with a large β still have better performance in terms of link utilization in the best case scenario. However, the effect of β becomes negligible when n becomes very large.

On the other hand, when there is a large number of flows sharing a link, the link is frequently overshot. Flows with a large α produce congestion more frequently. For instance, assume that k (of n) flows receive congestion signals when overshooting the link capacity r. The time duration to the next congestion moment is $r(1-\beta k/n)/(n\alpha)$. Since $k/n \ll 1$ when n is large, the congestion frequency is approximately proportional to α. Frequent congestion reduces the throughput of all flows, because they have to recover from more packet losses. Therefore, when the multiplexing is high, flows with a smaller α have higher efficiency in terms of link utilization. In summary, TCP-friendly AIMD flows with a large β and a small α have a higher bandwidth utilization, no matter whether the bottleneck link is highly or lowly multiplexed, as shown later in Section V-A2.

2) AIMD Responsiveness: Another property for congestion control is how quickly the AIMD-based algorithms respond to network and session dynamics, e.g., when the available bandwidth and number of flows change. The responsiveness benchmark that we use here for AIMD flows is *convergence*.

Convergence is generally measured by the speed at which the system approaches the steady state condition from any initial states [16]. For two AIMD flows, their cwnd traces converge to and oscillate in a line segment in steady state. If these two AIMD flows have the same (α, β) pair, this line segment coincides with their fair share line; or, if these two AIMD flows are friendly to each other, the converged line segment intersects the fair share line at its midpoint.

a) Single AIMD class: We first consider two flows from the same AIMD(α, β) class, F_1 and F_2, sharing a link with capacity r. F_2 takes the whole capacity before F_1 shares the link. Initially, their cwnd trace is at the point $(0, r)$, as shown in Fig. 4(a). After n decrease and increase *cycle*s, their cwnd trace is at the point $((1-\beta^n)(r/2), (1+\beta^n)(r/2))$. Since both flows have the same (α, β) pair, each *cycle* takes the same time, which equals $(1-\beta)r/(2\alpha)$ rtts. The convergence speed, denoted as c, is the speed of bandwidth gained by F_1, which equals the amount of bandwidth gained by F_1 over the time in rtts that it takes, i.e.,

$$c(\alpha, \beta) = \frac{\alpha(1-\beta^n)}{n(1-\beta)}. \tag{28}$$

From (28), the larger the α or β is, the faster the convergence is. However, if the (α, β) pair is under the constraint of the TCP-friendly condition in (13), we cannot increase both α and β simultaneously to get a higher convergence speed. By introducing (13) into (28), the convergence speed of TCP-friendly AIMD flow is

$$c(\beta) = \frac{3(1-\beta^n)}{n(1+\beta)}. \tag{29}$$

Therefore, the speed for F_1 to gain A portion of the total bandwidth is

$$c(\beta) = \frac{6A}{(1+\beta)\log_\beta(1-2A)} \tag{30}$$

where $0 < A < 0.5$. From (29) and (30), c, in packet per rtt^2, increases when β decreases. Therefore, when one-class TCP-friendly AIMD flows compete, the convergence speed is higher if β is smaller.

If the cwnd is a *continuous* variable, it will take infinite time for the cwnd trace to reach the steady state (when $A \to 0.5, \log_\beta(1-2A) \to \infty$). However, in reality, the cwnd is a nonnegative integer due to packetization, so within finite increase and decrease *cycle*s, the cwnd trace falls into the steady state, as shown in Fig. 4(b).

b) Multiple AIMD classes: Let two flows from different AIMD classes, $F_1(\alpha_1, \beta_1)$ and $F_2(\alpha_2, \beta_2)$, share the link, as shown in Fig. 5. Initially, their cwnd trace hits the capacity line at the point (x_0, y_0), where $x_0 + y_0 = r$. After one decrease and increase *cycle*, the cwnd trace is at the point (x_1, y_1), where x_1 and y_1 are

$$x_1 = \frac{r - \beta_2 y_0 + \frac{\alpha_2}{\alpha_1}\beta_1 x_0}{1 + \frac{\alpha_2}{\alpha_1}} \tag{31}$$

$$y_1 = r - x_1. \tag{32}$$

After this *cycle*, the bandwidth gained by F_1 is $(x_1 - x_0)$, and the number of rtt it takes is $(x_1 - \beta_1 x_0)/(\alpha_1)$. Then, the convergence speed in this *cycle* is

$$c; (\alpha_1, \beta_1; \alpha_2, \beta_2) = \frac{(r - \beta_2 y_0 - x_0)\alpha_1 + (\beta_1 - 1)\alpha_2 x_0}{r - \beta_2 y_0 - \beta_1 x_0}. \tag{33}$$

The partial derivatives of c with respect to α_1 and β_1 are both positive, while the partial derivatives of c with respect to α_2 and β_2 are both negative. It means for F_1 to gain the bandwidth faster, its own (α, β) should be larger, while the (α, β) pair for the competing flow should be smaller. Furthermore, if the (α, β)

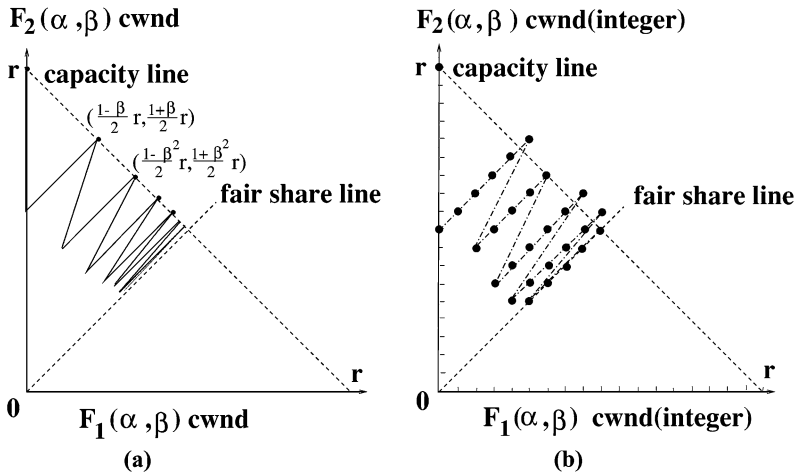

Fig. 4. Convergence of two AIMD flows from the same class: (a) cwnd is a continuous variable; (b) cwnd is an integer.

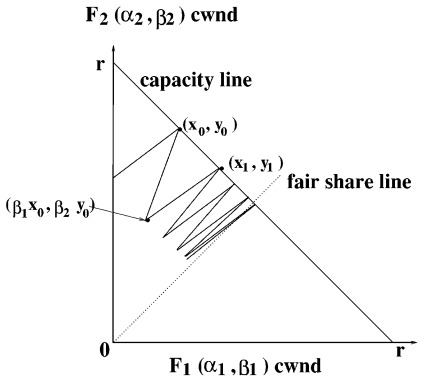

Fig. 5. Convergence of two AIMD flows from different classes.

of F_1 are under the constraint of the TCP-friendly condition (13), the convergence speed becomes $c(\alpha_1; \alpha_2, \beta_2)$. The partial derivative of $c(\alpha_1; \alpha_2, \beta_2)$ with respect to α_1 is always positive, which means c is larger if α_1 is larger and β_1 is smaller. The analysis of the bandwidth gain speed for the new arrival flow considers the worst situation that both flows throttle their cwnds simultaneously when they overshoot the available bandwidth. In reality, when the new flow has a much smaller cwnd than that of the old flow, it is likely that only the old flow suffers packet losses and shrinks its cwnd. In that case, the bandwidth gain speed for the new arrival is even faster.

In conclusion, for a new TCP-friendly AIMD flow to get its fair share bandwidth faster, its α should be larger, or its β should be smaller. Results in Section V-A3 confirm this observation. Although a new AIMD flow starts its competition by exponentially exploring the bandwidth through the Slow Start process, the above analysis is helpful to determine how fast the AIMD flow responds to the change of available network bandwidth.

C. Practical Implications

The achievable cwnd of an AIMD flow is determined by the available link capacity and rtt. When cwnd is very small due to the limited link capacity, the AIMD flow with a larger β has less throughput than the competing TCP flow, even its (α, β) pairs satisfy the friendly condition. In the following section, we discuss some practical implications that affect the TCP-friendliness of AIMD flows.

1) Sender Timeout: Timeout occurs when there are insufficient (less than 3) duplicated acknowledgments after a packet is lost. When timeout occurs, the TCP or AIMD flow will reinitialize its cwnd, and go through the Slow Start process. If timeouts occur simultaneously for both competing flows, their cwnd traces during Slow Start coincide with the fair share line. However, if timeout only occurs to one flow, that flow has a much lower throughput during its Slow Start than the other flow. Intuitively, timeout is less likely for flows with a large cwnd that have enough dupacks after a single packet loss. If cwnd is very small when congestion happens, timeout may occur and its effect on throughput should be further studied.

As shown in Fig. 1, when congestion occurs, the cwnd of AIMD flows is not equal to the cwnd of TCP flows in the *overload* region: the cwnd of the $\mathrm{AIMD}(\alpha, \beta)$ flow is $(3r)/(5+2\beta)$ and the cwnd of the TCP flow is $(2(1+\beta)r)/(5+2\beta)$. If β is larger than 0.5, the AIMD cwnd is less than the TCP cwnd in the *overload* region. Therefore, AIMD flows with a large β probably suffer more timeouts and thus have a lower throughput than TCP flows.

2) AIMD Parameters: The derivation of the friendly condition assumes that the cwnd is a continuous variable. In reality, due to packetization, the effective cwnd is always rounded to the maximum integer no greater than the algorithmic cwnd. When cwnd is small, the effect of this rounding is nontrivial. If cwnd is smaller than $1/(1-\beta)$ when congestion occurs, it has to be reduced at least by 1 packet to respond to the congestion signal although theoretically $(1-\beta)$ times cwnd is less than one. Therefore, the effective decrease ratio is $1 - 1/\mathrm{cwnd}$ which is smaller than β. When the available link capacity to each flow is so small that the average cwnd of each flow is smaller than $1/(1-\beta)$, the throughput of AIMD flows is much less than the analytical result. The larger the β is, the larger the $1/(1-\beta)$ is, and thus there is more negative effect on the throughput.

In addition, for TCP-friendly AIMD flows, if β is larger than 0.5, α should be less than 1. If there is no congestion, it takes $1/\alpha$ rtts to increase the effective cwnd by 1. During this time, if

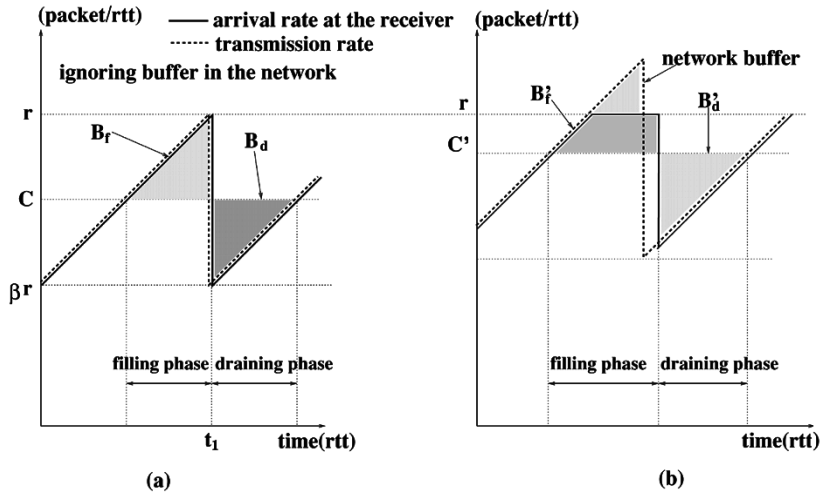

Fig. 6. *Filling* and *draining* phases: (a) without network buffer; (b) with network buffer.

a single packet loss occurs, the cwnd has no chance to increase before it actually decreases, i.e., the effective increase rate is less than α. If the cwnd is small when congestion occurs, the difference between α and the effective increase rate is nonnegligible. Therefore, AIMD flows with a large β and a small α may have a lower throughput than the competing TCP flows, especially when cwnd is small.

3) An Enhanced AIMD Algorithm—DTAIMD: To mitigate these practical implications, we propose a new Dynamic TCP-friendly AIMD (DTAIMD) algorithm. Since the effective decrease ratio, β', is $1-(1/\text{cwnd})$ when $\text{cwnd} < 1/(1-\beta)$, α should be adjusted based on β', instead of β, according to the TCP-friendly condition. When $\text{cwnd} \geq 1/(1-\beta)$, the increase ratio is the preset value α, which equals $(3(1-\beta))/(1+\beta)$; when $\text{cwnd} < 1/(1-\beta)$, the increase ratio should be $(3(1-\beta'))/(1+\beta')$, or $3/(2\text{cwnd}-1)$, i.e., α should be dynamically adjusted according to the current cwnd to ensure the practical TCP-friendliness.

The core pseudo code for the proposed DTAIMD algorithm based on the above analysis to enhance the TCP-friendliness of $\text{AIMD}(\alpha,\beta)$ flows is

```
L1      if   (cwnd ≥ 1/(1−β))      //if cwnd is large
L2           α = 3(1−β)/(1+β)      //regular condition
L3   elseif   (cwnd == 1)          //if cwnd is minimal
L4           α = 1
L5      else                       //if 1 < cwnd ≤ 1/(1−β)
L6           α = 3/(2cwnd−1)       //set w.r.t. effective β
```

If cwnd is large enough (i.e., $\text{cwnd} > (1/1-\beta)$), the regular TCP-friendly condition applies (L2). Otherwise, cwnd is adjusted according to the effective $\beta' = 1-(1/\text{cwnd})$(L6), unless cwnd is minimized already (L4). The algorithm is generic and adjusts α adaptively and dynamically, and it still preserves the desired AIMD properties. We will apply it to multimedia applications in Section IV. Later, we will evaluate its performance and compare it with the ordinary AIMD and TCP in Section V.

IV. DTAIMD-Based Multimedia Applications

In this section, we study how efficiently to apply the DTAIMD-based algorithms to support playback and multirate multimedia applications with service differentiation.

A. Multimedia Playback Applications

Web-based multimedia playback applications become popular in the Internet, and are advocated as the future of home entertainment. The stream length varies over a wide range, from a few second clips to hour length movies. Users desire the best affordable quality in terms of higher application throughput, minimum start-up latency, and less interruptions during playback. Let the DTAIMD-based protocol transport multimedia playback applications. Appropriate (α,β) pairs should be chosen to achieve the TCP-friendliness and better quality provisioning for these applications.

To simplify our analysis, let the playback application consume a stream at constant rate C. This assumption is reasonable as a starting point [18], and many schemes [19] have been proposed to smooth streams for easier resource reservation. These schemes can also be used in our context.

As shown in Fig. 6(a), if the packet arrival rate at the receiver (i.e., the transmission rate when the network buffering is ignored) is above C, the excess data are filled in the receiver buffer. This time period is referred to as the *filling* phase. At time t_1, packet loss occurs and the transmission rate is reduced below C; some data must be drained from the receiver buffer until the transmission rate reaches C again, and this time interval is referred to as the *draining* phase.

1) Low Multiplexing Scenario: Consider a static situation that only one AIMD flow traverses a link of capacity r. The transmission rate of the AIMD flow increases and decreases periodically. When the transmission rate reaches r, the cwnd decreases to βr. If the amount of data stored during the *filling* phase, B_f, is less than the data drawn from the buffer during the *draining* phase, B_d, the receiver will starve and the playback will freeze for a while, which is undesirable for playback

applications. To guarantee $B_f \geq B_d$, the consumption rate C should be bounded as follows:

$$C \leq (1+\beta)r/2. \tag{34}$$

From (34), the maximum consumption rate $C_{\max}$ of an AIMD(α, β) flow is proportional to $(1+\beta)$. The minimum receiver buffer size required (maximum queue length) is $[r^2(1-\beta^2)+6r(1-\beta)]/24$, i.e., with a larger β, the receiver buffer size can be reduced significantly, and so do the maximum delay and delay jitter. For instance, C of an AIMD(1, 0.5) flow is 25% less than that of an AIMD(0.2, 0.875) flow. A higher permissible consumption rate allows potentially higher application throughput and better audio or video quality. In other words, the AIMD protocol with a larger β can provide better QoS. However, if network buffer is considered, it has the effect of flattening the arrival rate to the receiver, shown in Fig. 6(b), and the advantage of the AIMD flow with a large β may become less significant.

2) High Multiplexing Scenario: When many AIMD flows with different (α, β) pairs compete for a highly multiplexed bottleneck, the $C_{\max}$ of a flow cannot be determined only by its own β as in (34). If the (α, β) pairs of competing flows satisfy the friendly condition, each flow has the same average throughput and $C_{\max}$. However, in a highly multiplexed dynamic network, AIMD flows with different (α, β) pairs have different response patterns to transient changes of network resources. AIMD flows with a small β and a large α are very sensitive to bandwidth variation, and their instantaneous throughput changes quickly. (It has been shown that the convergence speed of the TCP-friendly AIMD flow is inversely related to β in Section III-B2.) Therefore, during the playback, they are more likely to be interrupted. The frequency of interruption, or buffer underrun, is an important QoS index for multimedia playback applications, and the AIMD protocol with a larger β is desirable for these applications, as we will show in Section V-B.

B. Realtime Multirate Applications

The increasing demand for realtime video services has generated great research interests recently. Realtime multimedia applications are time-sensitive and cannot tolerate excessive delay and delay jitter. Therefore, many scalable multirate and layered video coding and decoding schemes are developed to cope with the variability of available bandwidth [21]. Given the resource available to them, heterogeneous receivers can tune into different coding layers to obtain the best quality.

Fig. 7 shows a multirate video flow under AIMD congestion control. When the AIMD sender increases its cwnd, the video source increases its coding rate accordingly. When there is a packet loss at time t_1, the AIMD sender reduces its cwnd from W to βW, and informs the video coder about this change. After an unavoidable delay τ, the video coder reduces the coding rate at time t_2. We are interested in the maximum sending delay since the packet is useless for realtime applications if its delay

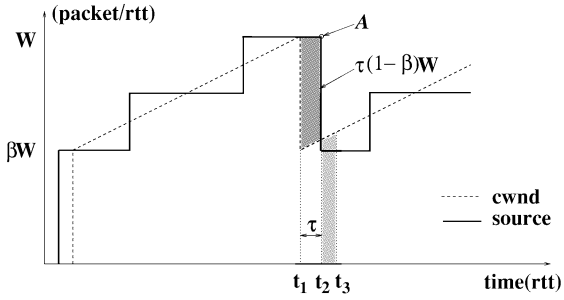

Fig. 7. AIMD for realtime multirate video applications.

exceeds a certain threshold. We focus on the queuing delay inside the sender. In Fig. 7, the packet $\mathcal{A}$ generated at time t_2 will suffer the longest sender queuing delay. During τ, the sending queue increases by about $\tau(1-\beta)W$ packets. d_q, the queuing time for packet $\mathcal{A}$ to be sent, can be calculated as follows

$$d_q \approx \frac{\tau(1-\beta)W}{\beta W} = \tau\left(\frac{1}{\beta}-1\right). \tag{35}$$

From (35), the maximum sender queuing delay, due to the mismatch between sending rate and coding rate, depends on the decrease parameter β. Let β be 0.875 instead of 0.5, the delay can be reduced by a factor of $6/7$, which is significant for realtime video.

In summary, choosing a large value of β has two advantages. First, it supports a scalable video coder to gracefully degrade user perceived quality when packet loss is detected. Secondly, the sending delay and delay jitter can be reduced considerably. On the other hand, under the TCP-friendly condition, the larger the β is, the smaller the α should be. Flows with a smaller α have lower convergence speed. Therefore, the application should make a trade-off to choose a suitable parameter pair according to its own traffic characteristics and QoS requirements.

C. Service Differentiation

Different multimedia applications can have different service requirements. To provide heterogeneous services, it is proposed to define several traffic classes and to allocate each class with a different priority of occupying network resources (e.g., bandwidth). If a family of AIMD parameters can be used to provide such differentiation, no major upgrades in the core network are required. It is a scalable and economical approach to incrementally deploy and offer service differentiation over the Internet.

From the analysis in Section III, the cwnd trace of co-existing AIMD flows must converge to a steady state. The average cwnds of two flows in steady state purely depend on their (α, β) pairs and are independent of link capacity. For two traffic classes, let class 2 have a higher priority than class 1, i.e., class 1 flows be weighted as 1 and class 2 flows be weighted as an integer $k > 1$. When they compete for the same bottleneck link, the throughput of the class 2AIMD(α_2, β_2) flows can be k times of that of the class 1 AIMD(α_1, β_1) flows if their average cwnd ratio is k.

Denote $t_i^{(1)}$ and $t_i^{(2)}$ the i-th time AIMD(α_1, β_1) and AIMD(α_2, β_2) in the *overload* region, respectively. $\Delta t_i^{(\cdot)}$ is

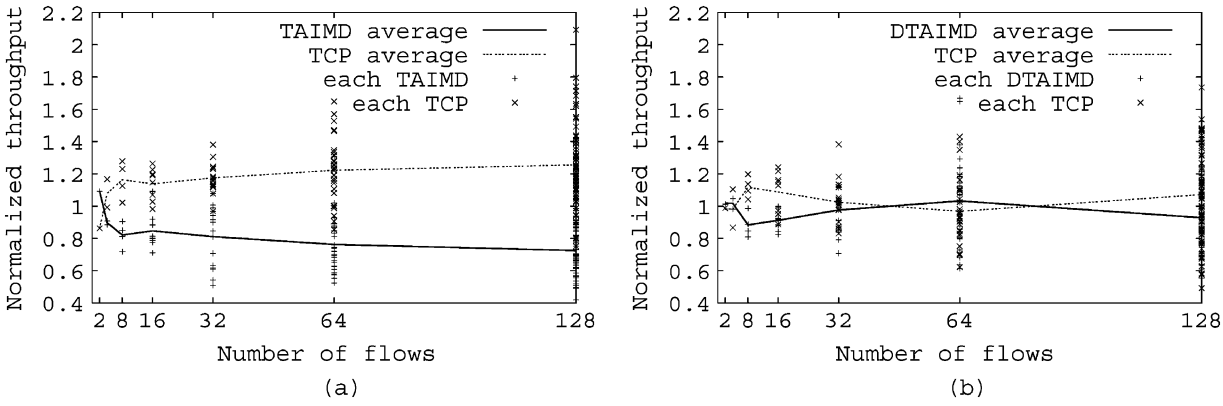

Fig. 8. Normalized throughput (RED, Link BW = 15 Mbps): (a) TAIMD(0.2, 0.875) versus TCP; (b) DTAIMD(0.2, 0.875) versus TCP.

the time duration between $t_{i-1}^{(\cdot)}$ and $t_i^{(\cdot)}$. Their cwnds are given by

$$W_1\left(t_n^{(1)}\right) = W_1(0)\beta_1^{n-1} + \alpha_1 \sum_{i=1}^{n-1} \Delta t_i^{(1)} \beta_1^{n-i} \quad (36)$$

$$W_2\left(t_n^{(2)}\right) = W_2(0)\beta_2^{n-1} + \alpha_2 \sum_{i=1}^{n-1} \Delta t_i^{(2)} \beta_2^{n-i} \quad (37)$$

From (36) and (37), we can get the average cwnds of two flows in steady state

$$E[W_1] = \frac{\alpha_1 E[\Delta t^{(1)}](1+\beta_1)}{2(1-\beta_1)}, \quad (38)$$

$$E[W_2] = \frac{\alpha_2 E[\Delta t^{(2)}](1+\beta_2)}{2(1-\beta_2)}. \quad (39)$$

The average sending rate of the AIMD(α_2, β_2) flow is k times that of the AIMD(α_1, β_1) flow. With RED-capable routers, the packet loss rate of AIMD(α_2, β_2) is also k times that of AIMD(α_1, β_1). In other words, the average duration of increase and decrease cycle of AIMD(α_2, β_2), $E[\Delta t^{(1)}]$, is $1/k$ of $E[\Delta t^{(2)}]$. Therefore, according to (38) and (39), to ensure $E[W_2] = kE[W_1]$, their (α, β) pairs should satisfy

$$\frac{\alpha_2}{\alpha_1} = \frac{k^2(1+\beta_1)(1-\beta_2)}{(1+\beta_2)(1-\beta_1)}. \quad (40)$$

For instance, let the data traffic be class 1 with $\alpha_1 = 1$ and $\beta_1 = 0.5$, and the multimedia traffic be class 2 with $\beta_2 = 0.875$. From (40), α_2 takes the values 0.8, 1.8, 3.2 for $k = 2, 3, 4$, respectively. With the (α, β) pair of (3.2, 0.875), a multimedia flow can have 4 times higher packet transmission rate than the co-existing data flows. In Section V-B3, we will illustrate this property with the simulation.

a) Implementation concerns: To achieve TCP-friendliness and service differentiation, applications should set (α, β) pairs according to their QoS requirements. However, to prevent abusive users or misbehaving applications, we suggest that an application can only choose one parameter according to its traffic characteristics. The operating system and protocol stack calculate the other one according to the TCP-friendly condition and the service differentiation weight. Obviously, this still cannot fully prevent malicious users from abusing the network by hacking the transport protocol or operating systems. The stability of the Internet relies on the voluntary cooperations from all end users. How to protect the network from malicious users is beyond the scope of this paper, since the DTAIMD protocol does not introduce any additional risks than the current TCP/IP stack.

V. Performance Evaluation

To validate the TCP-friendly condition derived in Section III-A, examine the bandwidth utilization and convergence speed analyzed in Section III-B, and compare with TCP the QoS provided by DTAIMD-based algorithms for multimedia applications detailed in Section IV, extensive simulations have been performed by using the Network Simulator (ns−2) [17]. The logical simulation topology is the widely used shared bottleneck topology. The simulations use the following parameters unless otherwise explicitly stated. The routers adjacent to the bottleneck link are RED-capable, and the link has 20 ms propagation delay. The number of competing flows ranges from 2 to 128, which covers the low multiplexing and high multiplexing (congestion) scenarios. To avoid the *phase effect* among competing flows, the rtt of each flow is made slightly different. MSS is 1000 bytes.

A. Performance of AIMD Algorithms

1) Flow Fairness: First, we evaluate the fairness among TCP, TCP-friendly AIMD (TAIMD), and DTAIMD flows in terms of *normalized throughput*. In Fig. 8, the x-axis represents the number of flows, and y-axis is the ratio of the flow throughput to their fair share. The normalized throughput should be 1 if the flow throughput equals the fair share, and the sum of the normalized average throughput is always 2 for two-class flows. Fig. 8(a) shows that TCP flows consistently have a higher average throughput than that of the TAIMD(0.2, 0.875) flows when they compete for the same link, especially when n is large and their average cwnd is small. This observation reveals the practical fairness problem between TCP and the ordinary TAIMD as we discussed in Section III.C. Fig. 8(b) shows that the normalized throughputs of TCP and DTAIMD(0.2, 0.875) are both close to 1, i.e., the average throughput of TCP and DTAIMD flows are close to each other, which implies better fairness when comparing to Fig. 8(a). This demonstrates that the proposed DTAIMD algorithm works

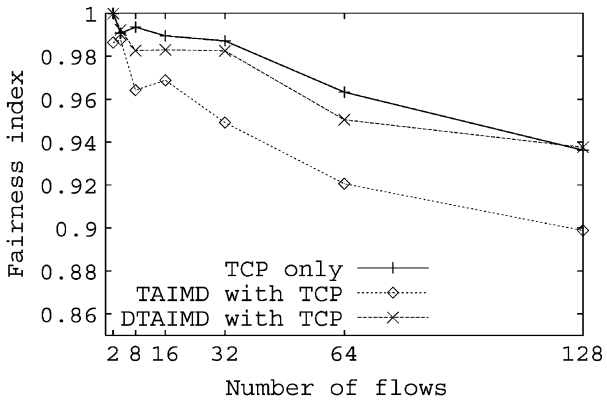

Fig. 9. Fairness index (RED, Link BW = 15 Mbps).

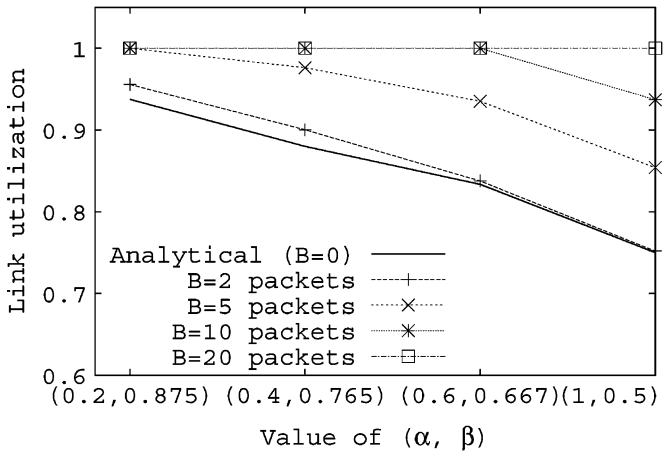

Fig. 10. Link utilization for one $\mathrm{AIMD}(\alpha, \beta)$ flow.

properly and does improve the fairness between the TCP and AIMD flows.

We further clarify this point by plotting the *fairness index* in Fig. 9 for TCP, TAIMD(0.2, 0.875), and DTAIMD(0.2, 0.875) flows. Fairness index is defined as $((\sum_{i=1}^{n} f_i)^2)/(n\sum_{i=1}^{n} f_i^2)$, where n is the number of total flows sharing the link and f_i is the throughput of the i-th flow. Fairness index has a range from $1/n$ to 1, and reaches 1 when all flows have the same throughput. Fig. 9 shows that the fairness index of DTAIMD(0.2, 0.875) flows with TCP flows is much better than TAIMD(0.2, 0.875) flows with TCP flows. It also shows that the fairness index of DTAIMD flows mixed with TCP flows is almost the same as that of TCP flows alone, which indicates that DTAIMD is truly TCP-friendly over a wide range of network and session dynamics. In the simulations, when there are more than 32 flows sharing the link of 15 Mbps, the average cwnd is less than three packets, and the congestions are mostly signified by timeouts. Simulation results show that the throughputs of DTAIMD flows are still close to those of TCP flows, which echos our statement that the derived TCP-friendly condition is valid even with frequent timeouts. Simulations with other TCP-friendly AIMD parameter pairs have the similar results, which are not presented here due to space limitation.

2) Link Utilization: To examine the link utilization derived in Section III-B1, four $\mathrm{AIMD}(\alpha, \beta)$ pairs, i.e., (0.2, 0.875), (0.4, 0.765), (0.6, 0.667), (1, 0.5), are chosen according to the TCP-friendly condition. Each simulation runs 30 s and the results are collected after 5 s from the beginning to avoid the skew introduced by the warming up effect.

a) Single AIMD flow: In this set of simulations, one AIMD flow traverses along a 3 Mbps link with a Drop-Tail queue. The rtt is around 40 ms. The buffer size is set to be two, five, ten, or 20 packets, respectively. Two is the minimum buffer size used in the simulation since the AIMD sender sometimes sends two packets back-to-back, e.g., when it receives an *ack* in Slow Start.

Fig. 10 enumerates four different $\mathrm{AIMD}(\alpha, \beta)$ pairs along the x-axis, and plots the bottleneck utilizations in the y-axis. It is shown that the simulation results match the analytical calculation. In addition, it shows that link utilization is proportional to β, and AIMD flow with a larger β needs less buffer to achieve a higher utilization.

b) Multiple AIMD flows: Now let two or four AIMD flows share the link. The buffer size is two, five, ten, or 20 packets for the two or four flows cases, respectively. Fig. 11(a) and (b) compare the link utilization with Drop-Tail and RED queues when there are four flows. When these flows share the link with the Drop-Tail queue, the link utilization with the minimum buffer size is very close to the analytical results of the worst case scenario, since all flows are synchronized when buffer overflows. However, the link utilization with the RED queue is very close to the analytical results of the best case scenario. It shows the RED-capable router can improve the link utilization by letting the competing flows react to congestion evenly in different phases.

We further increase the number of flows in the link to eight and 16. The minimum buffer size is five packets since it is very likely that more flows may simultaneously transmit two packets back-to-back and need more buffer to accommodate the traffic burst. The link utilizations with both Drop-Tail queue and RED queue are similar to the best case analytical results, as shown in Fig. 12(a) and (b). It reveals that when the link is highly multiplexed with a larger buffer size, the flows are not strictly synchronized as that in the worst case scenario, and higher link utilization can be achieved.

c) TCP versus TAIMD versus DTAIMD: To examine the link utilization for both low and high multiplexing scenarios when TCP flows compete with TAIMD or DTAIMD flows, the following five groups of flows sharing a 15 Mbps link are simulated: i) $2n$ TCP flows; ii) $2n$ TAIMD flows; iii) $2n$ DTAIMD flows; iv) n TCP flows and n TAIMD flows, and v) n TCP flows and n DTAIMD flows. n ranges from 1 to 64, which covers from the low multiplexing to high multiplexing scenarios. The RED queue thresholds are 25 and 125 packets.

As shown in Fig. 13, the link utilization is quite high when more than four TCP or DTAIMD(0.2, 0.875) flows share the link. The utilization is around 99% when there are more than four flows sharing the link. Fig. 13 shows that the link utilization remains high when TCP flows compete with DTAIMD(0.2, 0.875) flows. This fact indicates that $\mathrm{DTAIMD}(\alpha, \beta)$ is as efficient as TCP in terms of link utilization. Fig. 13(a) and (b) show that the link utilization is even better for TAIMD(0.2, 0.875) flows only or TAIMD(0.2, 0.875) flows mixed with TCP flows. This is because TAIMD(0.2, 0.875) has a large β and a small α, which is more efficient than TCP in link utilization no matter

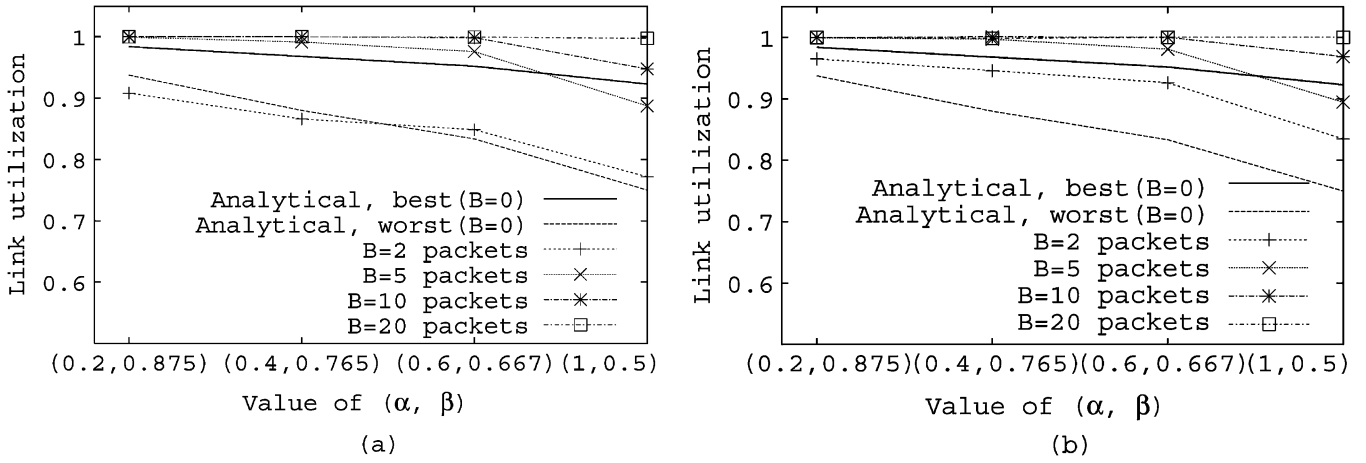

Fig. 11. Link utilization for four $\mathrm{AIMD}(\alpha, \beta)$ flows: (a) Drop-Tail queue; (b) RED queue.

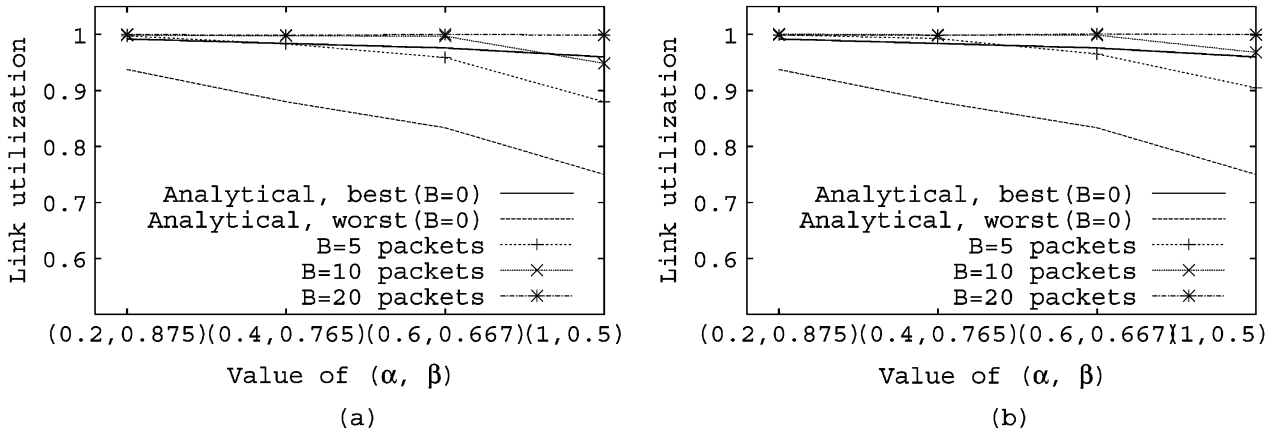

Fig. 12. Link utilization for eight $\mathrm{AIMD}(\alpha, \beta)$ flows: (a) Drop-Tail queue and (b) RED queue.

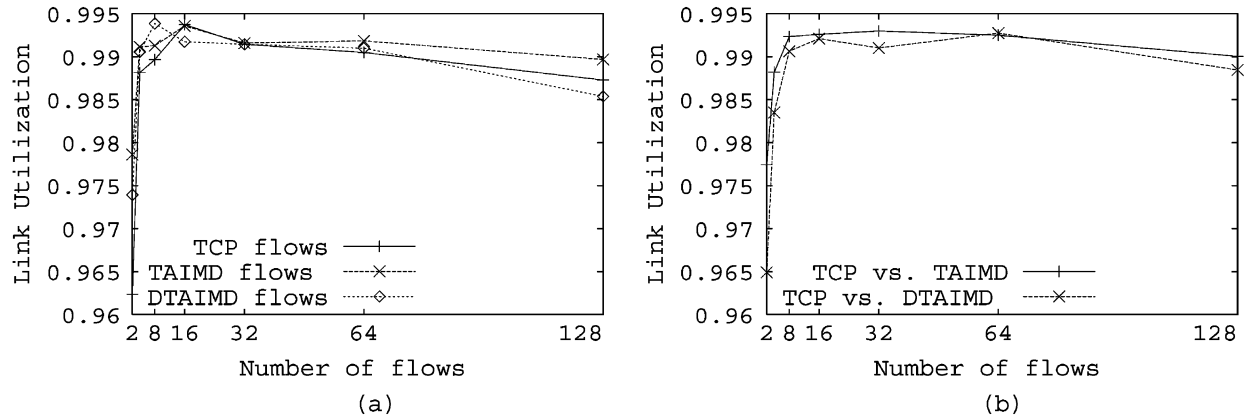

Fig. 13. Link utilization (RED, $\mathrm{Link\ BW} = 15$ Mbps): (a) homogeneous flows and (b) heterogeneous flows.

whether the link multiplexing is high or low, as shown in the discussions in Section III-B1.

3) Convergence Speed: Fig. 14(a) shows the results for the case with one AIMD class. The solid line is the analytical result for the convergence speed obtained in (30) and the dash line is the simulation result. Let one AIMD flow occupy the whole link at the beginning and the other one share the link later. The bandwidth gain speed of the second one is measured. To focus on the AIMD scheme, we freeze the Slow Start phase, and both flows increase and decrease their cwnds following the AIMD mechanism only. We set the link buffer to the minimum value; therefore, both flows receive the congestion signals once the sum of their sending rates exceeds the link capacity. The β for AIMD flows enumerated in the x-axis varies from 0.5 to 0.875, and α is set according to the TCP-friendly condition. Fig. 14(a) shows how fast a new AIMD flow gains 35% of total link capacity from an existing homogeneous AIMD flow. For TCP, or AIMD(1, 0.5), it is fairly quick to grab resources from competing flows. In addition, the convergence speed c drops quickly when β grows, and it confirms that c is inversely proportional to

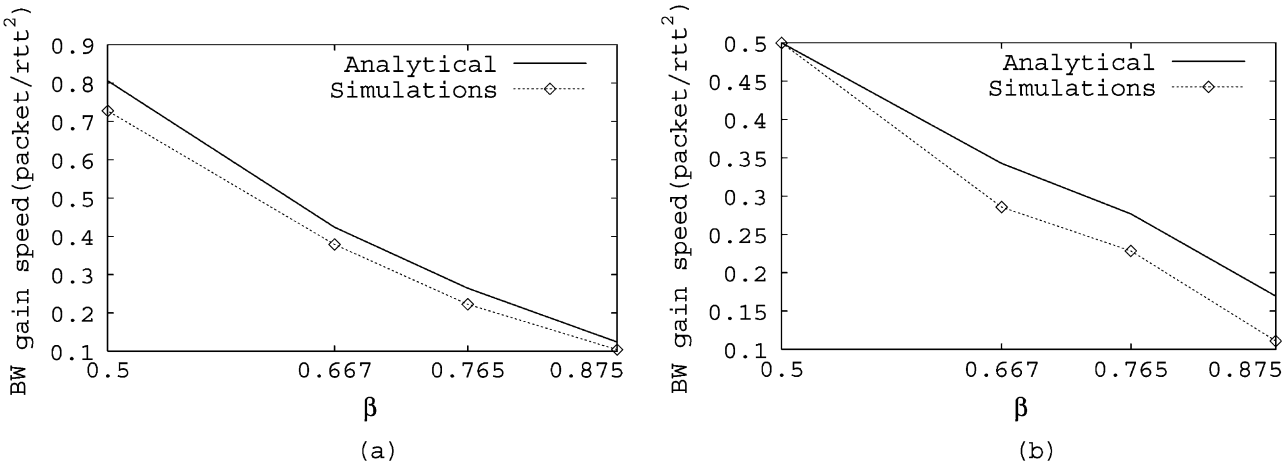

Fig. 14. Bandwidth gain speed for F_1, (link capacity 20 packets/rtt): (a) 35% bandwidth, one AIMD class; (b) first two cycles, two AIMD classes.

β, as derived in (29). In the analysis, we assume that the cwnds are continuous variables while in reality the cwnds are rounded when decreasing. Therefore, the simulation results are slightly lower than the c we derived.

Similar simulations have been done for multiclass AIMD flows. Fig. 14(b) shows the bandwidth gain speed for a new TCP-friendly AIMD flow competing with an existing TCP flow for the same bottleneck. The speed for the AIMD flow to gain bandwidth in the first two cycles is plotted in Fig. 14(b), since TCP gains most of its share in the first two cycles and its bandwidth gain speed approaches to zero after that. Since the AIMD flow rounds its cwnd when decreasing, the simulation results are slightly lower than the analytical one, especially when β_1 is much larger than the decrease ratio of the co-existing TCP. However, it reveals a similar trend that the larger the β is, the slower the convergence speed an AIMD flow has, as it becomes less aggressive to grab resources from others.

B. Performance for AIMD-Based Playback Applications

Fig. 15 shows the simulation configuration for multimedia applications: playback streams are stored as files in the server s_0 which is connected to a router r_1. The receiver d_0 is connected to the network through a router r_2. The target multimedia application shares network resources with other background traffic. In the current Internet, there are two kinds of dominant background traffic, *elephants* and *mice*. The *elephants*, such as the long-lived `ftp` connections, share the link with the target flow throughout the simulation. The *mice*, such as web transactions, usually have a very short lifetime. We consider both kinds of background traffic. *Mice* start randomly during the simulation and their lifetime follows an exponential distribution with a mean of 5 s.

1) Access Bottleneck: Let the connection bottleneck be the dedicated access line. Examples include a dialup modem connecting a home PC to ISP, or a wireless link connecting a PDA to the Internet. The access link bandwidth, d_0r_2, is 56 Kbps. There are ten cross traffic connections, which are *elephants*, sharing the r_1r_2 backbone link that has the capacity of 10 Mbps. In addition to the multimedia flow, there are n short-lived *mice* sharing the access link d_0r_2 and backbone link r_1r_2 with the target flow. n varies from 0 to 10. The packet size is 100 bytes for multimedia traffic and 1000 bytes for other traffic. DTAIMD(0.2, 0.875) and TCP are chosen to transport the multimedia flow separately. Fig. 16 compares their average throughputs. It shows that the average throughput of the DTAIMD(0.2, 0.875) flow is consistently higher than that of the TCP flow, i.e., DTAIMD(0.2, 0.875) can have a higher playback rate than TCP for multimedia applications, with changing background flows. Fig. 16 also indicates that the number of simultaneous connections in the access link should be minimized to achieve a higher playback rate for multimedia applications.

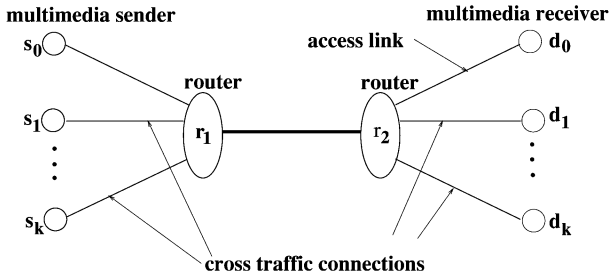

Fig. 15. Topology of the simulation with cross traffic.

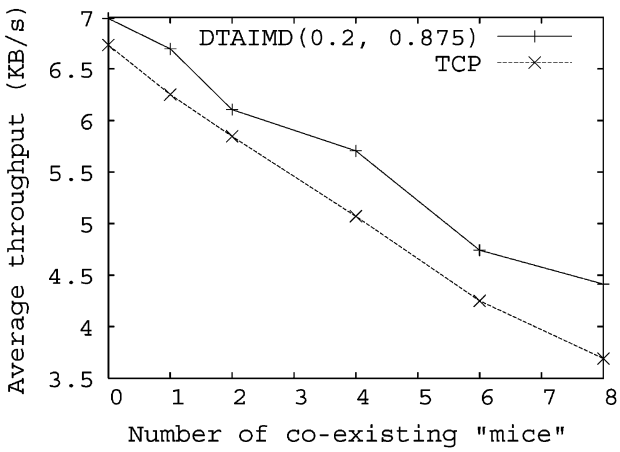

Fig. 16. Average throughput for the multimedia flow.

2) Backbone Bottleneck: Now let the connection bottleneck be the highly multiplexed backbone link. The multimedia receiver buffers data for several seconds before playback. (Several algorithms have been introduced in [20] to estimate

the playback start-up delay.) Ideally, the consumption rate equals the mean transmission rate. When the buffer becomes empty, the playback is interrupted and freezes for a while. In the simulation, one DTAIMD(0.2, 0.875) flow competes with 10 TCP *elephants* and 20 TCP *mice*, sharing the bottleneck link r_1r_2. The simulation lasts 600 s. The multimedia receiver of the DTAIMD(0.2, 0.875) flow initially stores the data for 3 s before it starts to playback at a constant rate, which equals its mean throughput. The DTAIMD(0.2, 0.875) flow and the TCP *elephants* are set to support multimedia playback applications separately. To compare the application performance transported by DTAIMD(0.2, 0.875) and TCP, the number of interruptions, the maximum queue length (KB) in the receiver end, and the average transmission rate (KB/s) of each flow are listed in Table I.

It shows that the DTAIMD(0.2, 0.875) flow can successfully playback the whole stream without any interruption, while 50% of the TCP flows have one to 12 interruptions. In the receiver end, the maximum queue length of the DTAIMD(0.2, 0.875) flow is less than that of 80% of the TCP flows. On the other hand, the average transmission rate of the DTAIMD(0.2, 0.875) flow is close to that of the TCP flows, which indicates that it is indeed TCP-friendly. By repeating the simulation several times, similar results are obtained. It implies that the DTAIMD(0.2, 0.875) protocol can have better QoS provisioning than the TCP protocol for supporting multimedia playback applications.

If without the random *mice* flows, DTAIMD and TCP *elephants* will reach a steady state, i.e., each flow additively increases and multiplicatively decreases its transmission rate periodically. When the random *mice* flows introduce disturbances to the network, TCP flows are more sensitive and have more prompt response to the disturbances than DTAIMD(0.2, 0.875) flows, as the convergence results shown in V-A3. Moreover, the number of TCP flows is much larger than that of DTAIMD(0.2, 0.875) flows, which is true in general. Therefore, the DTAIMD flow is immunized from the disturbances by the dominant TCP flows and can keep its flow smooth. On the other hand, although *mice* are more likely to grab bandwidth from TCP flows when they enter the network, TCP flows are more aggressive to grab it back after the *mice* leave the network. In a long term, the DTAIMD(0.2, 0.875) protocol can be TCP-friendly, and provide better QoS for multimedia playback applications.

3) Service Differentiation: We show the actual performance of utilizing the different $\mathrm{AIMD}(\alpha, \beta)$ pairs to offer service differentiation. Assume the multimedia flows (class 2) have a weight k over data flows (class 1). As derived in Section IV.C, their parameter pairs should satisfy (40). Fig. 17 shows the simulation results of the throughput ratio for multimedia flows over data flows with different k. The x-axis gives the total number of flows, and $1/6$ of them are multimedia flows. This figure is intended to show that the throughput of class 2 AIMD flows (for multimedia) is k times $(k = 2, 3, 4)$ that of the class 1 flows. Since there is no additional upgrade in the core network, different (α, β) pairs are very effective to offer service differentiation over the Internet, especially when flows are highly aggregated.

In summary, the TCP-friendly condition derived through the theoretical modeling in Section III offers satisfactory properties on TCP-friendliness, effectiveness, and responsiveness. Furthermore, the DTAIMD-based algorithm can efficiently support multimedia playback applications with service differentiation. These properties meet the design goals listed in Section I, and promise a scalable deployment over the Internet.

TABLE I
COMPARING TCP AND DTAIMD

	interruptions	max queue(KB)	tranx. rate(KB/s)
DTAIMD	0	1565	95
TCP-1	2	2350	91
TCP-2	12	3440	91
TCP-3	0	3132	88
TCP-4	1	1740	93
TCP-5	0	2329	91
TCP-6	0	1295	87
TCP-7	0	2111	82
TCP-8	7	1860	85
TCP-9	0	1731	84
TCP-10	5	1492	86

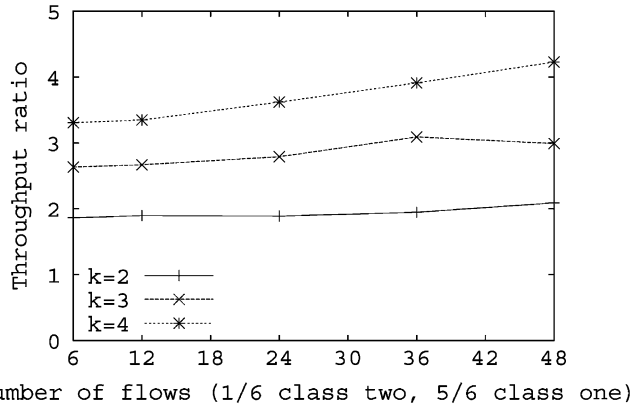

Fig. 17. Ratio of class 1 and class 2 average throughput.

VI. CONCLUSION

We have studied the TCP-friendly AIMD algorithms for multimedia playback and multirate applications. A novel approach has been used to derive the necessary and sufficient TCP-friendly condition directly, and to analyze the effectiveness and responsiveness of AIMD. Performance studies show that the DTAIMD-based protocol can be truly TCP-friendly over a wide dynamic range, and offer better QoS provisioning for playback and multirate applications. Since AIMD-based schemes are simple, compatible, and scalable, incremental deployment over the global Internet is feasible. The investigation on other TCP fairness issues like flows with different round-trip time and packet size, as well as some practical issues like fairness in a noncooperative environment, is under way.

APPENDIX

In the *underload* region, IIAD cwnd, W, increases as follows:

$$W(t+\Delta t) = W(t) + \frac{\alpha \Delta t}{W(t)\cdot \mathrm{rtt}},\ \alpha > 0. \qquad \text{(A.1)}$$

In the *overload* region, IIAD cwnd changes in the following way:

$$W(t^+) = W(t) - \beta,\ 0 < \beta < 1. \tag{A.2}$$

When an IIAD flow co-exists with a TCP flow in a link with capacity r packets per rtt, in steady state, the IIAD cwnd oscillates between $((\alpha-\beta)r)/(2\beta+r)$ and $(2\beta^2+\alpha r)/(2\beta+r)$, and TCP cwnd oscillates between $(r^2+(2\beta-\alpha)r-2\beta^2)/(4\beta+2r)$ and $(r^2+(2\beta-\alpha)r-2\beta^2)/(2\beta+r)$. Therefore, the ratio of average IIAD cwnd and TCP cwnd depends on r, i.e., no (α,β) pair can allow the IIAD flow to be TCP-friendly for arbitrary link capacity.

Let two IIAD flows, one has parameter pair (α_1,β_1) and the other has parameter pair (α_2,β_2), compete for a bottleneck link. In steady state, their average window ratio is

$$\frac{\overline{W_1}}{\overline{W_2}} = \frac{\alpha_2\beta_1^2 - 2\alpha_1\beta_2^2 + 2\alpha_1\beta_2 r - \alpha_1\beta_1\beta_2}{\alpha_1\beta_2^2 - 2\alpha_2\beta_1^2 + 2\alpha_2\beta_1 r - \alpha_2\beta_1\beta_2}. \tag{A.3}$$

Therefore, to achieve the ratio to be 1 and independent of r, the only solution is $\alpha_1=\alpha_2$ and $\beta_1=\beta_2$. In other words, for IIAD flows to be friendly with each other, only one (α,β) pair can be chosen for all flows.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for insightful comments and suggestions.

REFERENCES

[1] V. Jacobson and M. Karels, "Congestion avoidance and control," in *Proc. ACM SIGCOMM'88*, 1988, pp. 314–329.
[2] F. P. Kelly, "Stochastic models of computer communication systems," *J. Royal Statistical Soc.*, vol. B47, no. 3, pp. 379–395, 1985.
[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*, Mar. 1999, pp. 1337–1345.
[4] S. Karandikar, S. Kalyanaraman, P. Bagal, and B. Packer, "TCP rate control," *ACM Comput. Commun. Rev.*, vol. 30, no. 1, 2000.
[5] 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.
[6] Y. R. Yang and S. S. Lam. (2000, May) General AIMD Congestion Control. Univ. Texas, Austin. [Online]. Available: http://www.cs.utexas.edu/users/lam/NRL/TechReports/
[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] 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, Aug. 1999.
[9] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Comput. Commun. Rev.*, vol. 27, no. 3, 1997.
[10] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM'98*, 1998, pp. 303–314.
[11] S. Floyd and V. Jacobson, "Random Early Detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
[12] S. Floyd and T. Henderson, "The NewReno modification to TCPs fast recovery algorithm," in *IETF RFC 2582*, 1999.
[13] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment option," in *IETF RFC 2018*, 1996.
[14] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," in *IETF RFC 2481*, 1999.
[15] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.
[16] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. Comput. Networks ISDN*, vol. 17, no. 1, pp. 1–14, Jun. 1989.
[17] S. Floyd and S. McCanne. Network Simulator, LBNL public domain software. Available via ftp from ftp.ee.lbl.gov. [Online]. Available: http://www.isi.edu/nsnam/ns/
[18] R. Rejaie, M. Handley, and D. Estrin, "Quality adaption for congestion controlled video playback over the Internet," in *Proc. ACM SIGCOMM'99*, 1999, pp. 189–200.
[19] Z. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 429–442, Aug. 2000.
[20] R. Ramjee, J. F. Kurose, D. F. Towsley, and H. Schulzrinne, "Adaptive playout mechanisms for packetized audio applications in wide-area networks," in *Proc. IEEE INFOCOM'94*, 1994, pp. 680–688.
[21] H. M. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.

Lin Cai (S'00) received the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002. She is currently pursuing the Ph.D. degree in the same field at the University of Waterloo. She received a Postgraduate Scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC) in 2003. 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, ad hoc, and sensor networks.

Her research work has published in prestigious journals and conferences such as the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *ACM MobiCom*, and *IEEE Infocom*).

Xuemin (Sherman) Shen (M'97–SM'02) received the B.Sc. degree from Dalian Maritime University, China, in 1982 and the M.Sc. and Ph.D. degrees from Rutgers University, Piscataway, NJ, in 1987 and 1990, respectively, all in electrical engineering.

From September 1990 to September 1993, he was first with Howard University, Washington DC, and then the University of Alberta, Edmonton, AB, Canada. Since October 1993, he has been with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, where he is a Full Professor. His 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 ten 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 the *International Journal Computer and Applications*. He also serves as Guest Editor for IEEE JOURNAL ON SELECTED AREAS IN IN COMMUNICATIONS. *IEEE Wireless Communications*, and *IEEE Communications Magazine*. He 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. He is a registered Professional Engineer of Ontario.

Jianping Pan (S'96–M'99) received the B.S. and Ph.D. degrees in computer science from Southeast University, Nanjing, China in 1994 and 1998, respectively.

From 1999 to 2001, he was a Postdoctoral Fellow and Research Associate with the Centre for Wireless Communications, University of Waterloo, Waterloo, ON, Canada. From 2001 to 2003, he was a Member of Research Staff with Fujitsu Laboratories of America, Sunnyvale, CA. Since 2003, he has been a Research Scientist at NTT MCL, Palo Alto, CA. His research interests include protocols and applications for high-speed, multimedia, and mobile networks.

Dr. Pan is a member of the ACM.

Jon W. Mark (M'62–SM'80–F'88–LF'03) received the B.A.Sc. degree from the University of Toronto, Toronto, ON, Canada, in 1962, and the M.Eng. and Ph.D. degrees from McMaster University, Hamilton, ON, 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. In September 1970, he joined the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, where he is currently a Distinguished Professor of Emeritus. He served as Department Chairman from July 1984 to 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. He has been on sabbatical leave at IBM Thomas J. Watson Research Center, Yorktown Heights, NY, as a Visiting Research Scientist (1976–1977); AT&T Bell Laboratories, Murray Hill, NJ, as a Resident Consultant (1982–1983): Laboratoire MASI, Universite Pierre et Marie Curie, Paris France, as an Invited Professor (1990–1991); and Department of Electrical Engineering, National University of Singapore, as a Visiting Professor (1994–1995). 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 a text entitled *Wireless Communications and Networking* (Upper Saddle River, NJ: Prentice-Hall, 2003).

Dr. Mark is a Life Fellow of the 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).