Dynamic versus Static Traffic Policing: 
A New Approach for Videoconference 
Traffic over Wireless Cellular Networks

Polychronis Koutsakis

Abstract—The subject of traffic policing for computer communication networks has been studied extensively in the literature. However, the constant development of new multimedia applications which are “greedy” in terms of bandwidth and Quality of Service requirements calls for new approaches to the traffic policing problem. In this work, we introduce a new video model for single H.263 videoconference sources and we use it in order to propose a new traffic policing approach for wireless videoconference traffic. We study well-known traffic policing mechanisms which still present interesting, unsolved problems when servicing video traffic and propose, to the best of our knowledge, for the first time in the relevant literature that the token generator is based on a traffic model and not on a fixed rate. The proposed approach shows significant improvement in the results obtained by all the traffic policing mechanisms, and hence, shows that dynamic traffic policing can provide much higher efficiency than the widely used static approach.

Index Terms—Traffic policing, token bucket, traffic modeling, wireless videoconference, H.263 video encoding.

1 INTRODUCTION

Traffic from video services, especially videoconference traffic, is expected to be a substantial portion of the traffic carried by emerging wired and wireless networks [22], [42], [44], [45]. The explosive growth of wireless multimedia applications, in particular, calls for new sets of traffic control procedures to be implemented in order for the networks to cope with the bursty new applications, which have strict Quality of Service (QoS) requirements. For Variable Bit Rate (VBR) coded video, statistical source models are needed to design networks which are able to guarantee the strict QoS requirements of the video traffic. Video packet delay requirements are strict, because delays are annoying to a viewer. Whenever the delay experienced by a video packet exceeds the corresponding maximum delay, the packet is dropped, and the video packet dropping requirements are equally strict. As explained in [38], there are three areas where single video source models are useful: studying what types of traffic descriptors are needed for parameter negotiation with the network at call setup, testing rate control algorithms, and predicting the QoS degradation caused by congestion on an access link.

Hence, the problem of modeling video traffic, in general, and videoconferencing, in particular, has been extensively studied in the literature. In [39], various differences in successive video frame sizes of VBR video traffic were investigated. In [35], [36], [37], different approaches are proposed for MPEG traffic, based on the lognormal, Gamma, and a hybrid Gamma/lognormal distribution model. In [20], [21], the authors show that H.261 videoconference sequences generated by different hardware coders, using different coding algorithms, have gamma marginal distributions and use this result to build a Discrete Autoregressive (DAR) model of order one, which works well when several sources are multiplexed. This result was also employed by [41], which proposes an Autoregressive Model of order one for sequences of H.261 encoding.

Although H.264 [18] has been recently introduced as the new video coding standard, H.263 [11] is still very widely used in a variety of applications [15], [40], [46]. H.263 is a video standard that can be used for compressing the moving picture component of audiovisual services at low bit rates. It adopts the idea of PB frames, i.e., two pictures being coded as a unit. Thus a PB-frame consists of one P-picture which is predicted from the previous decoded P-picture and one B-picture which is predicted from both the previous decoded P-picture and the P-picture currently being decoded. The problem of modeling H.263 video traffic has been addressed in a few papers in the literature (e.g., [12], [33]). Videoconference traffic, however, has some inherent characteristics (e.g., very high autocorrelation) which differentiate the problem of its modeling from that of modeling video traffic.

In recent work [32], we have built a DAR model which accurately captures the behavior of multiplexed H.263 videoconference traces. We proceeded to use that model in our work in [9] to propose a new Call Admission Control mechanism for wireless cellular networks (i.e., a traffic control mechanism at the entrance of the system to prevent congestion, whereas the present work focuses on traffic control of users who have already entered the network). The DAR model was built based on extensive results showing that H.263 videoconference sequences have Pearson V marginal distributions. Since the Pearson V distribution is specified by two parameters which can be easily estimated from the mean and variance of the number of packets (cells)
per frame, only those two moments and the autocorrelation coefficient are needed to build a DAR model for multiplexed VBR videoconference sources. Our work for H.263 followed the steps of the aforementioned well-known work by Heyman et al. in [20], [21] for H.261 videoconference sources. The significant difference in our model was that we found that the gamma distribution, used in [20], [21] and also used widely in the past literature for MPEG-1 and MPEG-2 sources, does not provide the best fit for H.263 videoconference sources; instead, these are provided by the Pearson V (inverted gamma) distribution.

However, as shown in [34], [38], the DAR model fails to describe a single (i.e., not multiplexed with other sources) VBR source. The reason is that, due to the high autocorrelation of the video trace (in videoconference sources, autocorrelation is typically very high), the mean time between cell rate changes in the model is large, so the sample paths of the DAR model have "flat spots"; in the actual trace, however, the rate fluctuates (usually these fluctuations are small, but still they are frequent). This makes the DAR model inappropriate for traffic policing purposes, as each source needs to be monitored separately so that the system makes sure that the source conforms to its declared traffic parameters.

Hence, in this work, we use our results in [32] along with the work in [34] on the GBAR(1) model (which is discussed in Section 3 and is based on the Gamma distribution) in order to exploit the properties of the Pearson V distribution and be able to use a modified GBAR(1) model for single H.263 videoconference traces. We then proceed to utilize the model in our proposal for traffic policing of wireless videoconference sources, which is the main contribution of this work.

To the best of our knowledge, this is the first work in the literature which addresses the problem of modeling single H.263 videoconference traces. Very few other papers in the literature (e.g., [31]) have studied the modeling of H.263 traffic, but all of them have actually used low-rate versions of movies (i.e., not videoconference traces) for their work, and used the gamma distribution as the basis for their model.

In order to provide the required QoS guarantees, network resources need to be reserved according to both the QoS requirements and the specified traffic parameters of each application. On this subject, one of the fundamental network control issues is the source policing mechanism. The main goal of this control mechanism is to protect the network resources against intentional or unintentional traffic overflow from certain sources.

Several policing mechanisms have been proposed in the literature. Three of the mechanisms which have been most extensively studied are: the Token Bucket and its variations [2], [3], [4], [5], [7], [8], [10], [13], [14]; the Jumping Window [14], [16], [17]; and the Sliding Window (also known as the Moving Window) [14], [17], [19]. In this work, we implement in our system all three mechanisms, and propose several modifications to improve their performance, by using our videoconference traffic model as a token generation model. Our results show that our new approach of using the model for generating tokens provides better (in many cases, much better) results in policing the burstiness of video traffic sources than the static traffic policing mechanisms which have been widely presented and studied in the literature.

The rest of this paper is organized as follows: Section 2 presents our system model. Section 3 presents our new traffic model for single H.263 videoconference traces and discusses its accuracy. Section 4 presents the error model for the downlink (BS to wireless terminals) wireless channel. Section 5 contains: 1) a short description of the three well-known traffic policing mechanisms, along with their modifications which we propose in our study and 2) the evaluation of the results of our approach when incorporating our videoconference traffic model in all the traffic policing mechanisms considered. Finally, Section 6 presents the Conclusions of our study.

2 SYSTEM MODEL

Our study focuses on one cell (picocell) of the network. Within the picocell, spatially dispersed source terminals share a radio channel that connects them to a fixed base station (BS). The BS allocates channel resources, delivers feedback information, and serves as an interface to the mobile switching center (MSC). The MSC provides access to the fixed network infrastructure.

We consider a downlink (BS to wireless terminals) wireless channel, which is divided into time frames of equal length. Fourth generation mobile data transmission rates are planned to be up to 20 Mbps; therefore, in this work, we study a channel of this rate. Each frame has a duration of 12 ms and accommodates 566 information slots. Each information slot accommodates exactly one fixed-length packet that contains information and a header. In [30], it has been shown that the use in the uplink of a small portion of the channel bandwidth (less than 3 percent) for requests by terminals which wish to acquire slots to transmit is usually sufficient for high system performance. For this reason, we assume here that 16 of the 566 slots of the channel frame are used by the BS for acknowledgment of requests and transmission of synchronization information to the wireless terminals (these slots are noted as the control interval in Fig. 1); therefore, the available downlink channel bandwidth for information transmission (the information interval of the frame, shown in Fig. 1) is $550/566 \times 20$ Mbps = 19.43 Mbps. Hence, the information transmitted in the channel needs to be
restricted, by an efficient traffic policing mechanism, so as not to exceed this maximum available bandwidth.

We use computer simulations to study the performance of the traffic policing mechanisms. The simulations were conducted with the use of the C programming language. Each simulation point is the result of an average of 10 independent runs (Monte Carlo simulation).

3 Modeling single H.263 Videoconference Sources

The GBAR(1) process was introduced in [6] and was shown in [34] to be able to provide an excellent source model (named as the GBAR model) for single H.261 videoconference traces, thus alleviating the problem described in [19], [20] regarding the DAR model’s inability. In this work, we proceed to use the GBAR process in order to model single sources of H.263 videoconference traffic. In [32], we have used five different long sequences of H.263 encoded videos (from [1]) with low or moderate motion (as is the case in videoconference traffic), in order to derive a statistical model which fits well the real data. The length of the videos varies from 45 to 60 minutes and the data for each trace consists of a sequence of the number of cells per video frame. We have investigated the possibility of modeling each one of the five videos with quite a few well-known distributions; our results in [32] have shown that the use of the gamma distribution is not a good choice, whereas the Pearson V fit, in comparison to other distribution fits, has been shown with the use of powerful goodness-of-fit tests like Q-Q plots [20], [23] and Kolmogorov-Smirnov (K-S) tests [23].

The five traces used were, respectively:
1. A video stream extracted and analyzed from a camera showing the events happening within an office ("Office Cam").
2. A video stream extracted and analyzed from a camera showing a lecture ("Lecture Cam").
3. A video stream extracted and analyzed from a parking security camera ("Parking Cam").
4. A video stream extracted and analyzed from a talk show ("N3 Talk").
5. A video stream extracted and analyzed from another talk show ("ARD Talk").

For each one of these movies we have used the VBR coding version (in Quarter Common Intermediate Format (QCIF) resolution, which is a videoconferencing format providing a standard size for images produced by low-resolution digital cameras and video cameras; QCIF images are 176 pixels wide and 144 pixels tall). The trace statistics are presented in Section 3.2, where we validate our model for single H.263 videoconference sources.

Although the Pearson V is the best fit among all distributions, the degree of goodness-of-fit for the Pearson V varies for all traces. The reason that the Pearson V distribution fit cannot be a perfect fit in any of the examined cases is that the high autocorrelation between successive video frames in a videoconference trace can never be perfectly “captured” by a distribution generating independently frame sizes according to a declared mean and standard deviation, and therefore, none of the fitting attempts, as good as they might be, can achieve perfect accuracy.

This accuracy can, however, be achieved, with the use of a modified GBAR(1) [6], [34] model, which we propose in the following.

3.1 Model Definition

The GBAR(1) model is based on the Gamma distribution and some of its important properties. However, we are able to use the model taking advantage of the properties of the Pearson V distribution. More specifically, regarding the GBAR(1) model [6], [34], let Ga(β, λ) denote a random variable with a gamma distribution with shape parameter β and scale parameter λ. In this case, the density function is

\[ f_G(t) = \lambda e^{-\lambda t} \Gamma(\beta + 1), \quad t > 0. \]

Similarly, let Be(p, q) denote a random variable with a beta distribution with parameters p and q, and therefore, with density function

\[ f_B(t) = \frac{\Gamma(p + q) t^{p-1} (1-t)^{q-1}}{\Gamma(p+1) \Gamma(q+1)}, \quad 0 < t < 1, \]

where both p and q are larger than −1.

The GBAR(1) model is based on the well-known results that

1. the sum of independent Ga(α, λ) and Ga(β, λ) random variables is a Ga(α + β, λ) random variable
2. the product of independent Be(α, β − α) and Ga(β − α, λ) and Ga(β, λ) random variables is a Ga(α, λ) random variable.

Hence, if \(X_{n-1} = Ga(\beta, \lambda), A_n = Be(\alpha, \beta - \alpha),\) and \(B_n = Ga(\beta - \alpha, \lambda)\) and these three are mutually independent, then the following equation defines a stationary stochastic process \(\{X_n\}\) with a marginal distribution \(Ga(\beta, \lambda),\) and is called the GBAR(1) process

\[ X_n = A_n X_{n-1} + B_n. \]  

(1)

Given that the current value is determined by only one previous value, this is an autoregressive process of order 1.

The autocorrelation function of the GBAR(1) process is given by

\[ r(k) = (\alpha/\beta)^k, \quad k = 0, 1, 2 \ldots \]  

(2)

Assuming that the data have the property \(r(k) = \rho^k\) (which was found to be the case both for H.261 and H.263 videoconference traces), for \(k = 1, 2, \ldots, K,\) for some sufficiently large \(K,\) then, from (2), the parameter \(\alpha\) can be computed.

As explained in [34], the ability to simulate the GBAR(1) process only requires the ability to simulate independent and identically distributed gamma and beta random variables, which can be easily done, e.g., with methods from [23]. The parameters \(\beta\) and \(\lambda\) can be estimated from the mean and variance of the marginal distribution of the data from the actual video traces (the mean of the \(Ga(\beta, \lambda)\) distribution is \(\beta/\lambda\) and the variance \(\beta/\lambda^2\)).
PT5-distributed values

realization of the GBAR(1) model are used to get the

Pearson V distribution. This is possible because Z is a random

most accurate fit for all H.263 videoconference traces is not

to use the GBAR model although we found in [32] that the

generating noninteger values from (1) and then rounding to

end of the

frames arrive after 80 ms intervals, and

different video encoding schemes.

To validate our model, we use a generic buffer model, as

process. The values

expression:

The mean, peak, and standard deviation of the video

frame sizes for each of the five movies used in our work are

given in Table 1, along with the respective parameters of the

Pearson V distribution, computed from (3) and (4).

The Probability Density Function (PDF) of a Pearson V

distribution with parameters \((\alpha, \beta)\) is

\[
f(x) = \frac{x^{-\alpha+1}e^{-\beta x}}{\Gamma(\alpha)}, \quad \text{for all } x > 0, \text{ and zero otherwise.}
\]

The mean and variance are given by the following expressions:

\[
\text{Mean} = \beta/(\alpha - 1),
\]

\[
\text{Variance} = \beta^2/[(\alpha - 1)^2(\alpha - 2)].
\]

By computing \((\alpha, \beta)\) of the Pearson V from the above

equations, we get the respective parameters \((\alpha, 1/\beta)\) of

the Gamma distribution which we need for the GBAR(1)

process. The values \(X_n\) that we subsequently get by the

realization of the GBAR(1) model are used to get the

PT5-distributed values \(Z_n = 1/X_n\).

3.2 Model Validation

To validate our model, we use a generic buffer model, as

was done in [34], with the necessary modifications due to the

different video encoding schemes.

Video frames arrive in integer multiples of 40 ms

(P frames arrive after 40 ms intervals, PB frames arrive

after 80 ms intervals, and I frames arrive after 160 ms

intervals). Each frame contains a certain number of cells,

the exact amount of which is determined by either the data

trace or by a realization of the GBAR(1) model in order to

get the respective PT5-distributed values. All of the cells

have the same size (equal to 48 bytes in our work, but any

size can be chosen without any effect on our modeling and

traffic policing results). The cells are placed, if possible,

either in a buffer that can hold at most \(c\) cells or in an

infinite buffer (both cases were studied).

The cells are drained from the buffer at the rate \(d\) cells per 80 ms (the

80 ms interval was chosen since the vast majority of the

frames are PB frames). Let \(Z_i\) be the number of cells in the

\(i\)th frame and \(V_i\) be the number of cells in the buffer at

the end of the \(i\)th 80 ms interval. The timing is such that \(Z_i\) cells

arrive at time \(0\) and \(V_i\) is the number of cells present at time

80 ms. We assume that the buffer is empty at time \(0\), so

\(V_0 = 0\). The law of motion for \(V_i\) is

\[
V_i = \min[(V_{i-1} + Z_i - d)^+, \ c], \quad i = 1, 2, \ldots,
\]

where \(x^+ = \max(x, 0)\). Therefore, \(V_i\) is the smaller of two

numbers: the buffer capacity and the smaller of the buffer

content at the start of the \(i\)th interval, plus the cells which

arrived at the buffer, minus the cells which were drained

from the buffer (of course, \(V_i\) is always equal to the latter

number in the case of an infinite buffer).

Also, let \(Z_{avg}\) be the mean of any \(Z_i\); it is the arrival rate in

units of cells per 80 ms. The traffic intensity \(\tau\) is given by

\(Z_{avg}/d\) because \(1/d\) is the mean service time of a cell.

The mean, peak, and standard deviation of the video

frame sizes for each of the five movies used in our work are

given in Table 1, along with the respective parameters of the

Pearson V distribution, computed from (3) and (4).

The sets of parameters of these five traces comprise the

contract between the videoconference users and the wire-

less providers in our study (i.e., the user may have only one

of these possible contracts with the provider). This

approach is especially plausible for wireless videoconfer-

ence traffic, as the number of variations between source

bandwidth requirements is naturally restricted by the type

of application (a much larger pool of “contracts” would

have to be used in the case of video traffic). Since the focus

of this work is on the new proposal of the traffic policing

mechanism, we will present only indicative results of our

videoconference model validation.

Ten sample paths of the GBAR(1) process were gener-

ated and used as the arrival process (number of cells per

frame with a fixed interframe time which depends on the

frame type). The cell-loss rates from these paths were

averaged to obtain each point estimate for the GBAR(1)

model. The traffic intensity is varied by changing the

service rate \(d\).

Fig. 2 shows that the cell loss rates computed with the use

of the model are very close to the cell loss rates computed

from the actual data, regardless of the traffic intensity. These

results refer to the infinite buffer case. The reason that

the cell loss rate is highest in the case of lecture traffic is that

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Movie} & \text{Duration} & \text{Mean} & \text{Peak} & \text{Pearson V parameters} \\
\text{(minutes)} & \text{(bytes)} & \text{(bytes)} & \text{(bytes)} & (\alpha, \beta) \\
\hline
\text{Office} & 45 & 903 & 5191 & (9.623, 7793.128) \\
\text{Lecture} & 60 & 618 & 5760 & (4.787, 2338.862) \\
\text{Parking} & 60 & 2681 & 19680 & (30.568, 79288.216) \\
\text{N3 Talk} & 60 & 2545 & 13956 & (5.066, 10350.12) \\
\text{ARD Talk} & 45 & 2374 & 13275 & (5.352, 10331.12) \\
\hline
\end{array}
\]
lecture camera trace is by far the most bursty of all traces under study (burstiness = peak/mean = 9.32 for the lecture camera trace), and it has the highest standard deviation/mean ratio, which means that the values of the video frame sizes fluctuate significantly from the mean, hence adding to the bursty behavior of the trace. The reasons for the slight difference in the Parking trace curves in Fig. 2 are: 1) the very small, compared to all other traces, ratio of the standard deviation versus the mean, for this trace; this causes a much slower increase in the cell loss ratio for lower values of traffic intensity and 2) the fact that the Parking trace has, by far, the largest peak among all traces under study; this becomes a problem when traffic intensity increases, as the system is unable to cope with the large peaks when the channel is heavily loaded, and hence, the cell loss ratio increases abruptly.

Similar results to those in Fig. 2 were obtained when computing the mean queue lengths of the model and the actual traces in an infinite buffer, and when computing the cell loss ratio for the finite buffer case. As already explained, the quality of our modeling results is such that the model can be very effectively used in a new proposal for traffic policing of wireless videoconference traffic. This will be discussed in Section 5.

4 CHANNEL ERROR MODEL

We use a simplified Fritchman Markov model (from [29]) to emulate the process of packet transmission errors. The Markov model used in [29] for the downlink channel is presented in Fig. 3 and comprises of 15 states. State $s_0$ represents the “good state” and all other states represent the “bad states.” When the channel is in state $s_0$, it can either remain in this state or make the transition to state $s_1$ (with probability $p_0$). When the channel is in a bad state, the transition is either to the next higher state or back to state $s_0$, based on the status of the currently received packet. With this model, it is only possible to generate burst errors of at most length $N - 1$, where $N$ is the number of states (therefore, the maximum burst error in our model is 14 slots). The transition probabilities $(p_0, p_1, \ldots p_{13})$ of the error model are shown in Table 2.

The transition probabilities $(p_1, \ldots p_{13})$ of the error model have been adopted from [29]. The only difference in our model exists in probability $p_0$, which is, in our model, slightly smaller (0.00146) than in [29] (0.001469). The reason is that in [29], the probability that the downlink channel is in a good state is considered in the authors’ simulations to be 0.994. Given that in our work we assume a very strict QoS requirement of 0.01 percent maximum video packet dropping [27] (if a video packet fails to be transmitted before the arrival of the next video frame due to congestion or channel errors, the packet is dropped), the use of the 0.994 value for $p_{\text{good}}$ (i.e., $p_{s_0}$) would be prohibitive for our system to achieve the desired video QoS. Therefore, we have chosen in our study the value of the probability $p_{\text{bad}}$, i.e., the steady-state probability that the channel is in bad state, to be equal to $5 \times 10^{-5}$; this value has been chosen in order to test an “almost worst” case scenario for our system, given the maximum allowed video packet dropping.
probability of $10^{-4}$. Since $p_{good}$ in our model increases from 0.994 to 0.99995, $p_0$ needs to be slightly decreased, as is derived from the steady-state probability of the Markov chain. The average error burst length is 4.1 slots.

5 TRAFFIC POLICING BASED ON THE VIDEO TRAFFIC MODEL

In this section, we start by describing the three traffic policing mechanisms which we have used in our study, and the modifications we have introduced on them; we then present our respective results and discuss the efficiency of each mechanism in policing the videoconference traffic transmitted by the users over the channel.

5.1 The Token Bucket

5.1.1 The Function of the Standard Mechanism

The token bucket mechanism has been widely studied in the recent past. The reason for its popularity is its ability to verify easily whether a source conforms to its declared (at call setup) traffic parameters. The token bucket, along with the leaky bucket, the predominant methods for network traffic shaping. The two methods have different properties and are used for different purposes. The leaky bucket imposes a hard limit on the source transmission rate, whereas the token bucket allows a certain amount of burstiness (which is necessary for video traffic) while imposing a limit on the average source transmission rate [43].

The basic idea behind the token bucket approach can be described by the following:

- Tokens are put into the bucket at a certain rate. The bucket has a limited capacity.
- Each token represents a permission to the source to send a certain number of bytes into the network.
- After each transmission from the source, tokens which correspond to the packets transmitted by the source are removed from the bucket.
- Arriving packets of K bytes are conforming, and therefore, are immediately processed if there are tokens equivalent to K bytes in the bucket. If the current number of accumulated tokens (i.e., its equivalent in bytes) is less than the corresponding number of packets, the exceeding number of packets is nonconforming.
- Nonconforming packets either wait until the bucket has enough tokens for them to be transmitted (traffic shaping) or they are marked as nonconforming in order to be discarded in the case of network congestion or they are discarded, when their bandwidth needs exceed the token bucket size.
- If no packets wait to be transmitted, tokens can be accumulated up to the size of the token bucket. If the bucket fills with tokens and the source remains inactive or transmits at a rate lower than the token generation rate, the token buffer overflows and new incoming tokens are discarded, and therefore, cannot be used by future source packets. In this way, the token bucket mechanism imposes an upper bound on the source’s burst length, equal to the token bucket size, i.e., a token bucket permits burstiness, but bounds it. This bound can be

<table>
<thead>
<tr>
<th>Probability</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>0.00146</td>
</tr>
<tr>
<td>$p_1$</td>
<td>0.516068</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.778388</td>
</tr>
<tr>
<td>$p_3$</td>
<td>0.854118</td>
</tr>
<tr>
<td>$p_4$</td>
<td>0.936639</td>
</tr>
<tr>
<td>$p_5$</td>
<td>0.873529</td>
</tr>
<tr>
<td>$p_6$</td>
<td>0.905724</td>
</tr>
<tr>
<td>$p_7$</td>
<td>0.881041</td>
</tr>
<tr>
<td>$p_8$</td>
<td>0.831224</td>
</tr>
<tr>
<td>$p_9$</td>
<td>0.893401</td>
</tr>
<tr>
<td>$p_{10}$</td>
<td>0.863636</td>
</tr>
<tr>
<td>$p_{11}$</td>
<td>0.717105</td>
</tr>
<tr>
<td>$p_{12}$</td>
<td>0.853211</td>
</tr>
<tr>
<td>$p_{13}$</td>
<td>0.763441</td>
</tr>
</tbody>
</table>

Fig. 3. Channel error model.
described by the following formula: \( A(s, t) \leq \sigma + \rho(t - s), \ s < t \), where \( A(s, t) \) denotes the amount of traffic leaving the bucket between times \( s \) and \( t \), \( \sigma \) is the maximum burst size, and \( \rho \) is the token generation rate.

### 5.1.2 The Dual Token Bucket Approach

In our study, we used, at first, a dual token bucket mechanism in order to separately police both the peak and the mean rate of the source. Dual token buckets have been proposed and studied extensively in the literature (e.g., [13], [24], [26]). For the peak rate policer, the token generation rate was equal to the declared peak rate of the source, as was the token bucket size. For the mean rate policer, the token generation rate was equal to the mean rate of the source, but two methods of generating tokens were implemented. The first method was the widely used static method of generating tokens at a token rate equal to the mean (i.e., mean frame size divided by 80 ms), while the second method was our proposed method of generating tokens with the use of the Pearson V-based variant of the GBAR model for each video trace (tokens are generated by using the model, and put in the bucket every 80 ms; the parameters for the model are computed by the declaration of each source, at the system entrance, of its mean, peak, and standard deviation parameters).

Table 3 presents the loss (packets discarded as overload traffic and due to transmission errors in the wireless channel) when using each of the two methods, with various sizes of token buckets for each movie (size \( P \) denotes a bucket size equal to the declared peak frame size of the source). From the results shown in the table, we see that the loss experienced by all the traces is significantly lower (between 13.7 and 33.8 percent) when our proposed method is used, for all bucket sizes examined. The average decrease in packet dropping is 23.3 percent. The reason for this significant improvement is the quality of the video traffic model, which is able to “capture” the behavior of the video traces, including the variations in the traces’ bit rate, whereas the static mechanism generating tokens with a rate equal to the mean fails to do so. It needs to be stressed that in this study, we consider the very strict QoS requirement that a video packet is dropped if it fails to be transmitted before the arrival of the next video frame (and an upper bound of 0.01 percent is set on video packet dropping), since we want to test the system “at its limit” in order to study the efficiency of our dynamic traffic policing approach. However, depending on the video application and on buffer availability, the upper bound on packet delay can be more lenient, and in this case, the second policer will be responsible for the traffic shaping in the system, with a significant portion of the traffic shown as dropped in Table 3 being actually transmitted in that case, when tokens become available in the bucket.

These initial results showed that the idea of using probabilistic token generation instead of the widely used static one was very promising. As the bucket size increased, two facts became evident from the results in Table 3:

- video packet dropping is decreased and
- the difference between the static approach and our proposed method is also decreased.

The reason for both the above results is the same: as the bucket size increases, more tokens can be accumulated when small video frames are transmitted by the source, and the accumulated tokens help the system to overcome bursts of large video frames more smoothly. Therefore, video packet dropping decreases and the advantage of the use of the model slightly recedes. If we consider an infinite token bucket size, the packet dropping with both mechanisms would be zero, for “behaving” sources. However, using a large token bucket size defeats the purpose of policing, since it allows large successive bursts of video traffic to enter the network [7].

### 5.1.3 The Triple Token Bucket Approach

An important conclusion derived from the results in Table 3 is that, although our approach provides much better results than the static one, the video packet dropping results for both methods are unacceptably high; a strict QoS requirement for videoconference traffic sets the acceptable
percentage for packet dropping at just 0.01 percent (from [27]). It needs to be noted here that in all our results, the packet dropping enforced by the policers is added to the losses due to errors in the wireless channel (however, the latter are much smaller, due to the value of \( p_{good} \) in the channel error model, which is explained in Section 4). The very high packet dropping shown in Table 3 denotes the poor performance of the dual token bucket approach, as it is unfair to sources which conform to their declared traffic parameters. The reasons for this result will be discussed in more detail when we present our results on another policing mechanism in Table 6. Although the choice of having the token generation rate be equal to the source mean rate has often been used in the literature (e.g., [25]), in many other works (e.g., [8], [10]), the authors have reached the same conclusion as the one denoted by our results: if the token rate is equal or close to the source mean rate, well-behaved sources often have many cells marked as violating. Hence, three solutions can be applied to deal with the problem of high packet dropping from the second policer of the dual token bucket:

**First solution.** The bucket size can be considered infinite, which, as explained above, is a bad solution.

**Second solution.** Instead of a dual token bucket, a triple token bucket can be implemented. Triple token buckets have been used in the literature (e.g., [10], [28]), but not as often as dual token buckets. In both [10], [28], the triple token bucket mechanism uses one token bucket in series with two token buckets connected in parallel; the first token bucket in series controls the peak rate, and the system of the two parallel token buckets consists of one bucket which has a mean token generation rate larger than that of the source and a small token bucket size, and another bucket which has a mean token generation rate close to that of the source, but a large token bucket size.

On the contrary, in our proposed triple token bucket mechanism (shown in Fig. 4), all three buckets are connected in parallel and incoming traffic (video frames) needs to “pass” through each bucket.

The rationale behind our triple token bucket scheme is to create a mechanism which can be considered “fair enough” to video users so that if a percentage of the traffic is considered as violating by the third bucket, it will be immediately dropped at that hop.

More specifically, the first bucket behaves similarly to the first bucket in the dual token bucket scheme, policing (statically) the peak rate of the source with a token rate and bucket size equal to the peak. Traffic which fails to “pass” from this bucket is dropped. The second bucket (policing the mean rate of the source) also behaves similarly to the second bucket in the dual token bucket scheme, with one difference: source packets which fail to comply with the second policer are not discarded, but marked as nonconforming, and will only be dropped by the network in the case of network congestion (or, as already explained, in the case of a more lenient upper bound on packet transmission delay, this policer will be used for traffic shaping); otherwise, they are transmitted without any “penalty” being imposed on the source. The token rate (tokens are generated with the use of our model) is equal to the mean, and the bucket size for the second policer is chosen to be equal to 3P (i.e., three times larger than the peak video frame size of the source), in order to provide some system flexibility and less “marked” packets without allowing a large number of bursts from the source. The third bucket in our triple token bucket scheme is the one responsible for the source packets which are dropped, as it marks the “excess traffic” of the source as nonconforming. We set the token rate of the third token bucket equal to the average size of a “token frame.” A “token frame” is the average number of tokens needed to be
generated in each video frame for the source to lose only 0.01 percent of its packets (this size is found via simulation); we also set the token bucket size for the third bucket equal to 2P, in order to allow only a few successive bursts to “pass” this third policer. The reason is that the third policer is already very “tolerant”, with a token generation rate much higher than the average source rate. Traffic which fails to conform to the third policer is named as “excess traffic” of the source. Traffic which fails to conform to the first policer is also excess traffic, but the case of a source nonconforming to the peak policer is more straightforward and rare, and therefore, less important in this study.

The average number of tokens which need to be generated at the third policer in order to obtain a video packet dropping less than or equal to 0.01 percent is shown in Table 4 as a percentage of the peak traffic. A different upper bound in the video packet dropping rate would not influence the nature of our results regarding the studied traffic policing mechanisms. The different upper bound would only influence quantitatively the respective results presented in our tables; more specifically, in the case of the triple token bucket, the required token generation rate shown in Table 4 would be significantly smaller for a less strict QoS requirement (the values shown in Table 4 would be decreased by about 20 percent for a ten times larger upper bound of 0.1 percent in maximum video packet dropping).

The very significant differences between the percentages can be explained by the equally large differences in the standard deviation of each trace; the trace with the largest standard deviation in comparison to its mean (lecture camera trace) is the most “greedy” one in terms of required token generation rate, while the trace with the smallest standard deviation in comparison to its mean (parking camera trace) is the less greedy one. The greediness of sources with high standard deviation/mean ratio can be explained by the fact that the frame sizes of such sources have a larger variability than that of the frame sizes of sources with low standard deviation/mean ratio; this variability needs to be “covered” by the token generation rate of the third bucket, in order to keep the video packet dropping as low as 0.01 percent.

Regarding the practical implementation of the mechanism, it needs to be emphasized that it is quite simple due to the type of traffic considered. A logical assumption for next generation wireless networks is that videoconference users will be allowed to adopt one of a few specific “modes,” each corresponding to a set of traffic parameters. Therefore, by associating each one of the traces under study with a “mode,” it is easy to compute a priori the required token generation rate, shown in Table 4, for the third policer. This approach is especially plausible for wireless videoconference traffic, as the number of variations between source bandwidth requirements is naturally restricted by the type of application. A much larger pool of “modes” would have to be used in the case of video traffic.

The above discussion holds for the triple Jumping Window and Sliding Window mechanisms as well, as it will be shown in Sections 5.2 and 5.3.

Table 5 presents some indicative results (other possible combinations produce similar results) of our mechanism when a malicious user (i.e., a user attempting to violate its contract with the provider regarding its declared traffic parameters) is transmitting excess traffic over the network.

The efficiency of the mechanism is clear from our results, as the traffic which the malicious user attempted to transmit over the network is marked as nonconforming in its entirety (97-98 percent, in the two cases presented in the table), and a very large percentage of it (more than 35 percent in the two cases presented in the table) is severely “clipped” (dropped with the use of the first and third policer).

Third solution. We implemented again a triple token bucket, almost identical to the one described in the second solution, with one significant difference: instead of using a fixed token bucket size for the third policer, which is solely responsible for the video packet dropping in the absence of

<table>
<thead>
<tr>
<th>Table 4 Required Token Generation Rate at the Third Policer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movie</td>
</tr>
<tr>
<td>Parking</td>
</tr>
<tr>
<td>Office</td>
</tr>
<tr>
<td>ARD TALK</td>
</tr>
<tr>
<td>N3 TALK</td>
</tr>
<tr>
<td>Lecture</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5 “Clipping” a Malicious User: Percentage of Nonconforming Traffic in Each Policer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declared Movie</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Office</td>
</tr>
<tr>
<td>Lecture</td>
</tr>
</tbody>
</table>
network congestion, we allowed the bucket size to change dynamically over the simulation time (we only set an upper bound of $5P$ on its size, for the reasons already explained). According to this mechanism, changes in the third bucket size depend on the recent behavior of the source: if, within a fixed time window, the source transmits a small amount of traffic compared to its declared mean, there is a high probability that it will soon transmit information in bursts. Therefore, our mechanism increases the dedicated bucket size of the source, to help accommodate these anticipated bursts. Similarly, if the source transmits at a higher rate than its mean within the fixed time window, our mechanism decreases the dedicated bucket size of the source, in order to prevent a long overuse of network resources by malicious users.

We have implemented this dynamic mechanism with various upper and lower limits for the source generation rate, various time window sizes, and values for the bucket size. Although this triple token bucket mechanism is conceptually better than the one implemented in the second solution, due to its dynamic nature it did not produce better results (for some test values of source generation rate limits and bucket sizes, the results were better, but the decrease in video packet dropping was insignificant compared to the increased complexity of this mechanism). The reason that this mechanism does not produce better results is the very strict QoS requirement of videoconference traffic, which does not allow the flexibility of the dynamic mechanism to be fully exploited.

It should be noted that

- the token generation rate mechanism for the third policer is static. As the mechanism needs to generate tokens at a much higher rate than the mean of the source, there is no use in applying a model-based token generator.
- the dynamic triple token bucket mechanism (third solution) does produce better results than the static one (second solution) for large time windows (in the order of tens of frames); however, this is not a good choice, as the third token bucket is the most “tolerant” one and it should at least control burstiness, which is impossible if the rate control is not often performed. Therefore, we conducted all our simulation tests for this mechanism with relatively small time windows. In order to present a more complete study on the subject, however, we have studied the case of choosing large time windows (with a static third mechanism) in Section 5.2.
- all our results for the token bucket controlling the mean source generation rate show that, even with the significant improvement provided by our scheme, a high percentage of the source information will be marked as nonconforming and will be discarded in the case of network congestion. Therefore, a strict Call Admission Control mechanism must be implemented at the entrance of the network to prevent congestion.

### 5.2 Jumping Window

The Jumping Window mechanism uses windows of a fixed length $T$ side by side through time. A new window starts immediately after the conclusion of the previous one. During a window, only $K$ bytes (or packets) can be submitted by the source to the network. In the case that a source attempts to transmit more than $K$ bytes, the excessive traffic is dropped (or marked as nonconforming, as in the case of the Token Bucket). The mechanism is implemented with the use of a token counter, similar to the one of the Token Bucket, and in each new window, the associated packet counter is restarted with an initial value of zero [14].

**In our study, we use a modification of the Jumping Window mechanism, in order to implement a more dynamic traffic policing mechanism: in the case that less than $K$ bytes are transmitted by the source within one window, the token counter is not restarted, but starts with an initial value equal to the remaining tokens.**

Therefore, in our study, the jumping window mechanism can be regarded as an extension of a token bucket mechanism implemented over a longer time window than the ones studied in Section 5.1.

Similarly to the results presented in Table 3, Table 6 presents the loss for a source transmitting the N3 TALK trace when two Jumping Window mechanisms are used for policing. The peak rate is policed by a token bucket with generation rate and size equal to $P'$, where $P'$ denotes the peak amount of traffic transmitted by the source within one time window. We attempted to police the mean rate with various sizes of token buckets (as shown in Table 6) and with a generation rate equal to either the declared mean of the source, or a token generator based on our model.

The results presented in Table 6 show again clearly that the loss experienced by the trace is lower with the use of our proposed traffic policing scheme, for all bucket sizes examined (the results for the other four traces are similar in nature). However, the loss decrease with the use of the model here is smaller (both in percentage of the loss achieved with the static mechanism, and in absolute numbers) than the loss decrease in the case of the token bucket mechanism.

<table>
<thead>
<tr>
<th>N3 TALK</th>
<th>% Loss Static Mechanism (Tokens=Mean)</th>
<th>% Loss Proposed Method (Tokens=Model)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$P'$</td>
<td>$2P'$</td>
</tr>
<tr>
<td></td>
<td>21.8</td>
<td>20.6</td>
</tr>
</tbody>
</table>
shown in Table 3. The reason for this is the significantly larger window size (40 frames, i.e., 3.2 seconds); when studying the system at fixed, consecutive, but more widely dispersed time periods, the token generator based on the model naturally tends to behave similarly to the static one, as they have the same mean generation rate.

The comparison of the results in Table 6 to the respective results for the N3 TALK trace in Table 3 also shows that the loss for both the static and the dynamic mechanism is much lower with the use of the Jumping Window mechanism than with the token bucket. The reason is again the study of the system over a larger time window, within which a "behaving" source generates a traffic volume closer to the one predicted by its declared mean.

Similarly to the results presented in Table 4, Table 7 presents the token rate requirements when using a triple Jumping Window mechanism, similar to the triple token bucket mechanism. The bucket size for the second policer is \(3P^0\), and for the third policer \(2P^0\). The upper bound for the acceptable video packet dropping is again set equal to 0.01 percent.

The results of Table 7 show that the percentage of the peak traffic which needs to be offered to each trace as a token generation rate at the third policer in order to acquire a video packet dropping less than 0.01 percent is much lower for all the traces, compared to the respective percentage needed when the token bucket mechanism was used. The rationale of these results is the same as with the one explained in the discussion of the results of Table 6.

Fig. 5 presents a plot of the Jumping Window size versus the percentage of the peak traffic which needs to be offered to the office camera trace, in order to satisfy the packet dropping QoS requirement of the source. For small window sizes, this percentage decreases abruptly as the window size increases, while for large window sizes, the percentage tends asymptotically to 17.4 percent, which is equal to the mean/peak ratio of the office camera trace (i.e., for an infinite window size, a token generation rate equal to the mean of the source suffices for satisfying the very strict QoS requirement).

### 5.3 Sliding Window

The Sliding Window (Moving Window) mechanism is similar to the Jumping Window, but more stringent and more complex to implement. This mechanism again ensures that the maximum number of bytes transmitted by a source within any given time interval of duration equal to the fixed window size, \(T\), is upper bounded by \(K\) bytes.

The difference with the Jumping Window mechanism is that each video frame size is remembered for the width of exactly one window, starting with the specific video frame and ending \(T\) frames later. This mechanism can be interpreted as a window, which is steadily moving along the time axis, with the requirement that the frame sizes of \(T\) frames are stored for the duration of one window [14]. This is the reason that the implementation complexity is considerably higher than for the other two mechanisms (Token Bucket and Jumping Window), as the complexity is directly related to the window size; also, since the content of successive time windows differs by just one frame, it is clear that the

<table>
<thead>
<tr>
<th>Movie</th>
<th>Peak Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking</td>
<td>0.19</td>
</tr>
<tr>
<td>Office</td>
<td>0.2</td>
</tr>
<tr>
<td>ARD TALK</td>
<td>0.24</td>
</tr>
<tr>
<td>N3 TALK</td>
<td>0.32</td>
</tr>
<tr>
<td>Lecture</td>
<td>0.44</td>
</tr>
</tbody>
</table>
mechanism enforces the strictest bandwidth enforcement policy compared to the Token Bucket and the Jumping Window mechanisms.

Again, as in the case of the Jumping Window mechanism, we use, in our study, a modification of the Sliding Window mechanism, in order to implement a more dynamic policy: in the case that less than K bytes are transmitted by the source within one window $W$, the tokens left in the bucket are not discarded, but they are added to the token bucket of the next window ($W+1$), which has one more frames to police before it ends.

Tables 8 and 9 present the results for a dual and triple Sliding Window mechanism, respectively, similar to the dual and triple Jumping Window mechanisms.

The Sliding Window size has been chosen equal to the Jumping Window (40 frames) for the purpose of comparison. With the use of this mechanism, we observe from Table 8 that the packet loss results for the dual policer are higher than those achieved with the Jumping Window mechanism (Table 6). The reason for this result is the significantly large window size; in our discussion of Table 6, when the Jumping Window mechanism was used, we explained that when the system is studied at fixed, consecutive, but more widely dispersed time periods, the token generator based on the model naturally tends to behave similarly to the static one, as they have the same mean generation rate. Also, as already mentioned, when the Sliding Window policer is used instead of the Jumping Window one, the consecutive periods (windows) differ by just one video frame. The combination of a large window and a frame-by-frame progress enforce the very strict nature of the Sliding Window mechanism, in comparison to the Jumping Window. The results for the rest of the traces are similar to those presented for the N3 Talk trace in Table 8.

Table 9 presents the token rate requirements when using a triple Sliding Window mechanism, similar to the triple token bucket and triple Jumping Window mechanisms. The bucket size for the second policer is again $3P'$, and for the third policer, $2P'$. The upper bound for acceptable video packet dropping is again set to 0.01 percent. The results of Table 9 show that the percentage of the peak traffic which needs to be offered to each trace as a token generation rate at the third policer in order to obtain a video packet dropping less than or equal to 0.01 percent is much lower for all the traces, compared to the respective percentage needed when the token bucket mechanism was used, but much larger than the percentage needed when the Jumping Window mechanism was used. The rationale for these results is the same with the one explained in the discussion of the results of Tables 6 and 8.

6 Conclusions
In this work, we have proposed a new traffic policing approach for videoconference traffic transmission over wireless cellular networks. We first built a new model for single H.263 videoconference sources, and then proceeded to use the model in order to improve, with several modifications that we proposed, the standard traffic policing mechanisms used in the Token Bucket, Jumping Window, and Sliding Window methods. To the best of our knowledge, this is the first work in the relevant literature using an adaptive probabilistic token generation rate based on a traffic model, instead of a fixed rate. Our results show that our dynamic approach provides significantly better results in policing the burstiness of video traffic sources than the static traffic policing mechanisms which have been widely presented and studied in the literature.

The dual token bucket scheme was shown to work rather poorly based on our results for both the classic approach and ours; therefore, we proposed a triple token bucket scheme, which showed the best performance. The reason we adopted the triple token bucket scheme is that in the case of the dual token bucket (where a very high percentage of video packets is unable to “pass” the second bucket), a choice would have to be made on whether this traffic would be dropped or if it would be simply “marked” as nonconforming and dropped later inside the network, if congestion was detected (this choice is again not optimal, as

<table>
<thead>
<tr>
<th>Movie</th>
<th>Peak Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parking</td>
<td>0.2</td>
</tr>
<tr>
<td>Office</td>
<td>0.49</td>
</tr>
<tr>
<td>ARD TALK</td>
<td>0.51</td>
</tr>
<tr>
<td>N3 TALK</td>
<td>0.64</td>
</tr>
<tr>
<td>Lecture</td>
<td>0.68</td>
</tr>
</tbody>
</table>

TABLE 9
Required Token Generation Rate at the Third Policer for the Sliding Window Mechanism
a large portion of the marked traffic would be dropped, although this may not be needed for the decrease of network congestion). With our triple token bucket scheme, the above problem is resolved; the situation can be considered “fair enough” to video users so that if a percentage of the traffic is considered as violating by the third bucket, it will be immediately dropped at that hop. This choice, combined with the use of the video traffic model in the second bucket for characterization of the rest of the violating traffic as nonconforming, still allows the network to drop the violating traffic in case of congestion; however, it has first “helped” decrease the possible congestion by dropping a part of this traffic and this decreased congestion probability means that it is more probable that the rest of the nonconforming traffic will not be dropped later inside the network. Therefore, video QoS will adhere to the desired QoS requirements.

Hence, in next generation wireless cellular networks, where the large number of multimedia users will call for strict traffic policing, a triple-policer scheme can provide a very competent solution to the problem. Based on our results when studying the Token Bucket, Jumping Window, and Sliding Window schemes, we believe that in the case when all users are considered equal by the provider (i.e., in the case that all users are charged equally for a given service), the better choice is the triple Jumping Window mechanism which is the more “lenient” of the three in terms of “clipping” videoconference sources which generally conform to their declared traffic parameters but, at times, have large bursts due to the nature of video traffic. If, on the other hand, the provider offers multiple possibilities of charging to users depending on their willingness to pay for better service, then the triple Jumping Window should be used for the most high-paying users, while either the triple Token Bucket or the triple Sliding Window mechanism should be used for low-paying users (the triple Token Bucket mechanism is less complex to implement, but is even stricter than the Sliding Window mechanism).

One key issue which needs to be addressed with respect to our traffic policing proposal is the applicability of the scheme as a more general proposal for next generation networks. Since the scheme is focused on H.263 encoded video, it could be argued that the applicability of the scheme is restricted to the specific traffic and its wide variety of applications. However, the importance of the proposed scheme is related to the fact that videoconference applications are very greedy in terms of bandwidth requirements. Data traffic is much less bandwidth-consuming; therefore, it is much easier for the system to accommodate. The same stands for other types of traffic, such as voice and streaming video, which have the additional advantage of limited or no burstiness, in comparison to data traffic. Therefore, it is actually not necessary to adopt the same approach for the policing of all types of flows in the network.

Our traffic policing approach is also applicable for the new video coding standard, H.264. In our latest work [47], we have shown that traffic from multiplexed H.264 videoconference sources can be modeled with DAR(1) based on the Pearson V distribution, as H.264 videoconference sequences have Pearson V marginal distributions. The only difference with H.263 is that the modeling is more complex due to the fact that I, P, and B frames need to be separately modeled for H.264 sources, whereas H.263 traces can be modeled as a whole. Since all of our results confirm that the idea of using probabilistic token generation is very promising, the next step in our work will be to implement the modified GBAR(1) model to single H.264 videoconference sources and study the implementation of our proposed traffic policing approach for that type of videoconference traffic.

ACKNOWLEDGMENTS

The author wishes to thank the Associate Editor and the anonymous reviewers of the paper. Their remarks and suggestions have significantly helped toward enhancing and better explaining this work. This work was conducted while the author was with the Electrical and Computer Engineering Department, McMaster University, Hamilton, Ontario, Canada.

REFERENCES


Polychronis Koutsakis received the five-year diploma in electrical engineering in 1997 from the University of Patras, Greece, and the MSc and PhD degrees in electronic and computer engineering from the Technical University of Crete, Greece, in 1999 and 2002, respectively. He was a visiting lecturer in the Electronic and Computer Engineering Department at the Technical University of Crete for three years (2003-2006). From July 2006 till December 2008, he was an assistant professor in the Electrical and Computer Engineering Department at McMaster University, Canada, and his research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC). In January 2009, he re-joined the Electronic and Computer Engineering Department at the Technical University of Crete as an assistant professor. His research interests focus on the design, modeling, and performance evaluation of computer communication networks, especially on the design and evaluation of multiple access schemes for multimedia integration over wireless networks, on call admission control and traffic policing schemes for both wireless and wired networks, on multiple access control protocols for mobile satellite networks, wireless sensor networks, and powerline networks, and on traffic modeling. He has authored more than 80 peer-reviewed papers in these areas, served as a guest editor for an issue of the ACM Mobile Computing and Communications Review, served as TPC cochair for the 14th ACM WMUNEP 2008, and serves as a TPC member annually for most of the major conferences in his research field.