A Dynamic TCP-Aware Call Admission Control Scheme for Generic Next Generation Packet-Switched Wireless Networks

Xinbing Wang, Do Young Eun, and Wenye Wang

Abstract— **Traditional Call Admission Control (CAC) schemes only consider call-level performance and are mainly designed for circuit-switched wireless network. Since future wireless communications will become packet-switched systems, the packet-level features could be explored to improve the system performance. This is especially true when the TCP-type of elastic applications are running over such packet-switched wireless networks, as the elasticity of TCP applications has more tolerance toward the throughput/delay variation than non-elastic traffic does. In order to efficiently utilize the system resource from an admission control perspective, we propose a TCP-aware CAC scheme to regulate the packet-level dynamics of TCP flows. We analyze the system performance under realistic scenarios in which (i) the call holding time for non-elastic traffic like voice is independent of system states and (ii) the call holding time for TCP type of traffic *depends* on the system state, i.e., on the TCP flow's transmission rate. Extensive simulations are presented under different scenarios to show that the proposed scheme can effectively improve the system performance in terms of call blocking probability, call-level throughput (call/min) and link utilization, in accordance with our theoretical results.**

Index Terms— **Admission control, wireless networks, TCP.**

I. Introduction

With the enormous success of TCP in wired networks such as Internet [1], [2], it is assumed to be responsible for data transmission over a wireless environment [3] at least for the near future, and extensive research has been conducted to make the TCP transmission over wireless networks feasible. Examples include TCP with ACK and Window regulators [4], [5], ATCP [6], M-TCP [7], I-TCP [8], etc. As a consequence, TCP flows over wireless links become more robust against unreliable wireless channels, e.g., the packet loss ratio being smaller than 1% [4], [5]. Hence, it is reasonable to expect that TCP will continue to play a key role in data transmission over the future wireless networks.

A salient feature of the TCP congestion control is so-called the Additive-Increase-Multiple-Decrease (AIMD) algorithm. If there is no packet loss or ECN mark within the current round trip time (RTT), the window size is increased by one during the next RTT; Otherwise, the window size is halved. While the existing literature regarding TCP over wireless networks varies in different aspects, they share a common TCP congestion control algorithm, i.e., AIMD, which affects the packet-level TCP dynamics and the associated performance for a packet-switched network.

As the packet-switched technology offers more efficient utilization of the scarce wireless resource than that of circuit-switched networks, the future 3G and 4G wireless network will be packet-based ones [9], [10] and this is especially true for 4G systems.[1] In a packet-switched wireless network, the packet-level dynamics is central to the call-level performance, and thus both the packet-level and the call-level have to be considered in Call Admission Control (CAC). Traditionally, CAC schemes in wireless networks deal only with call-level performance metrics: call blocking probability and handoff dropping probability [11], [12], [13], [14]. Recently, CAC schemes in [15], [16] take the packet-level performance into account. In particular, an efficient CAC scheme was proposed in [16] for heterogeneous services in wireless ATM networks, where the packet-level constraints such as delay and delay-jitter are used for predefined QoS. A guard-channel based idea has been widely applied to the CAC scheme [14], where a higher priority is assigned to handoff calls to improve the handoff dropping probability by reserving certain amount of bandwidth for potential handoff calls. Their simulation results also indicate that the proposed CAC scheme can achieve both the packet-level and the call-level performance in terms of the link utilization and handoff dropping probability.

However, none of these schemes has considered the effect of TCP applications on the packet-level performance. As TCP is largely responsible for carrying data over the future packet-switched wireless network, it is important to investigate the effect of TCP algorithm on CAC. Intuitively, the TCP AIMD algorithm makes the TCP *elastic* to fit in the network automatically – if a network has more capacity, TCP flows keep increasing the packet transmission rate into the network until there is a packet loss. On the other hand, a packet loss detected by the TCP flow gives rise to a reduction in the TCP injection rate into the network. Compared to non-elastic traffic like voice, TCP type of elastic traffic is insensitive to the packet level dynamics [17], [18] in that it is more tolerant toward the throughput/delay variations than non-elastic traffic does.

Manuscript received January 26, 2006; revised September 9, 2006 and December 27, 2006; accepted January 11, 2007. The associate editor coordinating the review of this paper and approving it for publication was M. Zorzi.

X. Wang is with the Dept. of Electronic Engineering, Shanghai Jiaotong University, China (email: xwang8@sjtu.edu.cn).

D. Y. Eun and W. Wang are with the Dept. of Elec. and Comp. Engineering, North Carolina State University, Raleigh, NC, USA (email: {dyeun, wwang}@eos.ncsu.edu).

Digital Object Identifier 10.1109/TWC.2007.06064.

[1]We note that 3G is a circuit and packet switched network, but 4G will be a pure packet-switched network.

This feature allows us to intensionally control the elastic TCP throughput for the benefit of desirable system performance.

Specifically, in order to fully utilize the elasticity of TCP applications over a packet-switched wireless network, we first layout a two-level framework which allows us to investigate CAC over the future packet-switched wireless networks. Based on the framework, we then propose a TCP-aware CAC scheme that takes the effect of AIMD algorithm into consideration. The idea is to have the system intentionally control the packet loss rate of TCP calls based on the current system status, rather than allowing the TCP calls to arbitrarily increase the injection rate to the point of system saturation and buffer overflow (the buffer can be per-flow [19] or aggregate buffer). In particular, if the system is under light load (i.e., the number of calls in the system is small), the system will ‘choose’ smaller packet loss probability of TCP calls to achieve higher average TCP throughput. (TCP throughput is roughly inversely proportional to the square root of the packet loss probability.) In contrast, if the system is under heavy load (i.e., the number of calls in the system is large and the system capacity is nearly fully-utilized), the system will impose higher packet loss probability on TCP calls to reduce their throughput (it can be achieved through packet dropping or Explicit Congestion Notification (ECN) marking [20]), and this allows the system to accommodate more call requests in order to reduce the call blocking probability. We analyze the system performance under the following realistic assumptions: (i) the call holding time for non-elastic traffic is independent of system states, and (ii) the call holding time for TCP type elastic traffic *depends* on the system state, i.e., on the TCP flow’s packet transmission rate. We proceed extensive simulation results showing that by dynamically controlling the packet loss probability of the TCP calls, we can effectively regulate the system so as to yield the desirable performance metrics both at the call level and at the packet level. In addition, the performance gain in our scheme is shown to remain the same regardless of the call holding time distributions, indicating that our scheme can be applied to a wide range of situations.

II. Preliminary: Elasticity of TCP AIMD Algorithm

The main feature of TCP congestion control algorithm is characterized by TCP AIMD [21]. Specifically, assuming that the window size of a TCP source in the current round trip time (RTT) is W. If all the W packets are transmitted successfully to the destination, the TCP source will inject $W+1$ packets in the next RTT. If at least one of the W packets is lost, the TCP source will reduce the window size to $W/2$. In this paper we assume that packet loss due to the wireless link is handled by physical or MAC layer such that TCP source can distinguish whether the loss is due to congestion (i.e., buffer overflow) or wireless links [6], [7]. Without a congestion signal, the TCP sources will keep increasing the rate injected to the network until a packet loss occurs and the capacity of a base station (BS) will be fulfilled sooner or later. The arrival packets, when the buffer is full, will be dropped by the BS and the lost packets become the congestion signal for the TCP sources to reduce their sending rates. In this situation, the average throughput of a TCP call can be expressed by the well-known formula [22], [23]

$$\mathbb{E}\{TCP\} \approx \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}}, \tag{1}$$

where $p(t)$ is the packet loss probability. This equation shows that the average TCP throughput can be determined by $p(t)$ for a given RTT.

With the advances of 3G wireless technology [19], it is known that the packet drop rate p_i can be reduced within 1% through intelligent channel state scheduling, and link layer retransmission mechanisms (See [4], [5] and the references therein). According to the specifications of CDMA-EVDO systems [19], the base station deploys a buffer for each admitted flow and uses a scheduler to allocate radio resources among all the on-going flows. Such a buffer mechanism allows the system to control the packet drop rate p_i for each flow by intentionally dropping the packets within the corresponding buffer.

III. System Model

We study a generic packet-switched wireless network, which could be GPRS system, CDMA-EVDO system [19], or some other type [9], [10], supporting multiple classes of services as in [24], [25]. Unlike traditional CAC schemes for wireless networks that only take call-level performance metrics into account, we look at a system with two-level considerations: the call-level and the packet-level.

At call-level, the BS will allocate radio resource to the call request–either new call or handoff call request. Each class of calls is characterized by its call arrival rate, and service holding time, etc., while the packet-level features are characterized by the QoS profile that describes the packet arrival rate, packet queueing delay and packet loss ratio requirements. Different classes may have different packet-level QoS requirements. For example, the voice traffic requires much smaller delay than data traffic does, and the non-real time traffic usually has more stringent loss requirement than the realtime traffic.

The interaction between the call level and the packet level is the following. If a BS accommodates more call requests into the system and each request has certain amount of data to be transmitted, the overall system load in terms of packets (average packet arrival rate) will increase, which is the ‘action’ of the call level on the packet level. Hence, at the packet level, the system with fixed capacity (bits/sec) is more likely to be overwhelmed by the increase in the number of calls. In other words, the probability of packets dropped due to buffer overflow is increased. Since each type of service has certain QoS constraints (e.g. the packet loss probability being less than some predefined value), in order not to violate the QoS agreement with ongoing users, the system may throttle the admission of an arrival request and this is the ‘reaction’ of the packet level to the call level.

Remark: Our approach is generic and can be applied to general packet switch wireless networks. It can also be tailored to the specific wireless system with proper modifications. For example, our scheme can be applied to the wireless system like GPRS, where multiple users can send their data

TABLE I
SYSTEM PARAMETERS: CALL-LEVEL AND PACKET-LEVEL.

Call-Level	
K	number of classes, $k = 1, 2, ..., K$
Λ_k	call arrival rate of class k (call/min)
Ψ_k	call service rate of class k (call/min)
P_b^k	call blocking probability of class k
Packet-Level	
C	system capacity (bits/sec)
$\tilde{C}$	QoS guaranteed capacity
$\overline{p}$	QoS guaranteed packet loss prob.
p_T^u	upper bound of TCP packet loss prob.
p_T^l	lower bound of TCP packet loss prob.
$p(t)$	packet loss prob. of TCP call

whenever there is some room to obtain the multiplexing gain for higher channel utilization. As to 3G wireless networks (say, CDMA-EVDO [19]), one buffer is allocated to each flow. In this situation, we can still apply our scheme with suitable modifications. In an EVDO system, if a TCP flow fills its own buffer and overflow occurs, the system would interpret it as an indicator that the system is near the saturation, and the system cannot admit the new call request any more. Meanwhile, such a per-flow buffer scheme allows the system to conveniently regulate the throughput of each TCP flow by intentionally dropping the packets in each corresponding buffer. Overall, we point out that our approach is not limited to 3G packet switched wireless networks (e.g., UMTS) and in fact can be applied to any generic packet switched environment (e.g., 4G system).

In order to quantify the system behavior, we summarize system parameters in Table I that will be used in the rest of the paper.

IV. TCP-AWARE CAC SCHEME

In this section, we present our TCP-aware CAC scheme and describe how the packet-level dynamics affects the admission control at call level. Generally, the QoS profile for each class of calls is fixed. For example, the packet arrival rate of voice traffic is fixed and the packet loss probability of voice traffic should be less than a certain threshold. If the system serves only those classes of calls with a fixed QoS profile, the system does not have any flexibility to deal with (either accept it with guaranteed QoS, or reject it). However, for a TCP call, the QoS requirement is not fixed, but falls into a wide range, as TCP is a self-adaptive protocol. If there is more bandwidth available, TCP will increase the transmission rate. If there is little bandwidth left in the system and congestion happens (or is likely to occur), TCP will reduce the transmission rate to fit in the available bandwidth. Thus, TCP itself is capable of coping with a wide range of situations and this feature allows the BS to have the flexibility to control the TCP calls. For example, suppose that there is a new call request and the system does not have enough resource to admit this call (e.g., the total packet arrival rate is near the capacity). Then, instead of rejecting the call immediately, the system will try to reduce the transmission rate of TCP calls by controlling the packet loss probability, $p(t)$ (cf. (1)) of TCP calls to make some room for the new call request. This increases the chance of accommodating the new call request and thus decreases the call blocking probability. Next, we illustrate how our TCP-aware CAC works in detail.

Algorithm 1 TCP Aware CAC: New Call Admission.

```
Require: Base Station (BS) status
 1: A new call request 'm' arrives
 2: if BS has enough resource then
 3:    Accept 'm'
 4: else
 5:    "TCP Call Adjustment"   /* adjust TCP calls */
 6:    if BS has enough resource after TCP adjustment then
 7:       Accept 'm'
 8:    else
 9:       Reject 'm'
10:    end if
11: end if
```

Algorithm 2 TCP Aware CAC: Dynamic TCP Call Adjustment at Packet Level.

```
Require: $p(t)$, $p_T^U$, $\tilde{C}$, QoS profile of call 'm', $E_m$ is the average
    transmission rate of call 'm'.
 1: $n_1 \leftarrow$ the number of TCP call in the system
 2: $\lambda \leftarrow$ the total packet arrival rate
 3: for i = 1 to $n_1$ do
 4:    /* reduce TCP call's transmission rate */
       $\Delta\lambda \leftarrow \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}} - \frac{1}{RTT}\sqrt{\frac{3}{2p_T^U}}$
 5:    $\lambda \leftarrow \lambda - \Delta\lambda$
 6:    if $\lambda + E_m \leq \tilde{C}$ then
 7:       Accept 'm'
 8:       Return "True"
 9:    end if
10: end for
11: Reject 'm'
12: Return "False"
```

When there is a new call arrival [2] in the system, the BS will examine whether there is enough resource or not, i.e., whether the total packet arrival rate is smaller than the 'QoS guaranteed capacity' $\tilde{C}$ or not ($\tilde{C}$ is obtained in Section V-A). If it is, the BS will accept the request; If not, the BS will do 'TCP call adjustment' (see Algorithm 2). After TCP call adjustment, if the system has enough resource, the new call request will be accepted, otherwise, the call will be rejected. This is the algorithm for new call admission as shown in Algorithm 1. We note that Algorithm 1 mainly deals with call-level decision, and the packet-level consideration is reflected in Algorithm 2: TCP call adjustment, where the BS will enumerate all the ongoing TCP calls and reduce the packet-level transmission rate of TCP calls. Hence, the total transmission will be decreased until there is enough room to accommodate the new request. If there is no enough capacity to accept this call after the BS enumerates all the TCP calls, the call request will be rejected. In addition, like TCP traffic, the UDP is treated as elastic traffic in our scheme and the system simply drops any excess packet based on the system status and only the minimum QoS as specified in the QoS profile will be guaranteed.

[2]In order not to deviate from our objective, we do not consider handoff calls in this paper.

A. Discussion on $p(t)$ and RTT in Real Implementation

The TCP throughput is subject to two key parameters as shown in (1): one is p(t) and the other is RTT, both of which may change over time, rather than remain constant all the time. As the TCP throughput heavily depends on the expression in (1), a realistic question is how the BS can *accurately* measure the throughput of each TCP flow in practice? Or, equivalently, how can the BS accurately estimate the $p(t)$ and RTT?

In reality, we know that each TCP flow maintains its own timer and counter to keep track of RTT, the number of packets sent out, and the number of packets acknowledged, hence is aware of its own packet loss rate (say, $p'(t)$). The BS is also able to maintain the packet loss rate $p(t)$ since it intentionally drops the incoming packets for each TCP flow with probability $p(t)$. The BS may obtain this $p'(t)$ information from the TCP sender, and calculate the error $|p(t) - p'(t)|$. If the error is within some tolerable range δ, then the BS knows that the packet loss probability experienced by the TCP sender is indeed what it observes, i.e., the number of dropped packets divided by the number of packets passing through the base station. If the error is larger than the tolerable range, then the BS will simply choose the largest value of the packet loss probability to ensure that the aggregate packets will not overflow the packet level system capacity. We display the algorithm in Figure 1.

```
Algorithm: Obtain $p_i(t)$
Input: The inquiry period $T$; the tolerable error $\delta$;
       The BS has the initial $p_i(0)$.
Output: The BS obtain the accurate $p_i(t)$.
Method:
   01. The BS measures incoming packet rate for flow $i$.
   02. Every $T$s, the BS send $p_i(t)$ request to flow $i$.
   03. Each TCP flow returns the average $p'_i(t)$ to the BS.
   04. IF $|p_i(t) - p'_i(t)| \le \delta$
   05.     The error within the tolerable range.
   06.     $p(t+1) \leftarrow$ updated packet loss probability.
   07. ELSE
   08. The BS updates its own $p_i(t)$ from flow $i$.
   09.     $p(t+1) \leftarrow \max\{p'_i(t), p_i(t)\}$
   10. END
```

Fig. 1. The algorithm for the base station to obtain $p(t)$.

Remark: By introducing this $p(t)$ into the BS for rate control, we may bring some overheads in the computation, design, and implementation. Intuitively, this $p(t)$ is unlike the power control, error control or signaling protocols, which are all essential features already embedded in the BS. In order to implement the algorithm in Figure 1 in practice, we have to consider the following issues. First, the BS needs to keep track of the transmission rate for each TCP flow. Second, it should periodically communicate with each TCP flow to obtain the RTT information from each flow. A closer look at both of these overheads, however, reveals that the actual overhead can be kept minimal and well within the range of future wireless system capacity. As to the computational overhead, note first that the computational overhead for each flow is fairly small as the expression in (1) is very simple. Further, the number of flows in a cell in the future wireless cellular system tends to be smaller in order to increase the wireless bandwidth utilization (e.g., micro cell, pico cell, etc.), ranging from tens to several hundreds at most. Considering the fast increasing computational power of DSP, we can reasonably expect that the computational overhead will not be an obstacle. On the other hand, since the mobile terminals will keep updating their status information to the base station frequently, the $p(t)$ and RTT information can easily be implemented through piggy-pack along with the existing exchange information between the BS and the TCP senders.

V. Performance Analysis

In this section, we analyze the system performance of call blocking probability under the following assumptions: (i) for non-elastic traffic like voice etc: the service holding time is independent of the transmission rate as commonly assumed in most of the existing CAC literature [14], [13]; (ii) for elastic traffic like TCP data flow: the service holding time is *dependent* on the transmission rate. Assumption (ii) is very general and can be applied to most data transmission situations, where the higher the transmission rate, the shorter the service time. For a fixed file size, if the corresponding transmission rate of TCP call is larger, then the total service time for the file transfer will be smaller, and vice-versa.

A. Call Level State Space

Define $x = (x_1, ..., x_K)$, where x_k is the number of calls of class k. Assume TCP call is the class 1 and the average throughput of TCP call $\mathbb{E}\{TCP\}$ is given by (1). Let $\mathbb{E}\{X_k\}$ be the average packet-level throughput of class k ($k = 2, ..., K$), which is determined from the predefined QoS profile. Let Ω be the set of all the admissible/possible states in the system and $\Omega_k \subset \Omega$, ($k = 1, ..., K$) be the set of all the 'boundary states' that class k call will be blocked. We formulate Ω and Ω_k as follows.

$$\Omega = \left\{ x \in \mathbb{Z}^{+K} \,\middle|\, x_1 \mathbb{E}\{TCP\} + \sum_{k=2}^{K} x_k \mathbb{E}\{X_k\} \le \tilde{C} \right\}, \quad (2)$$

and based on Ω, we can get the subset Ω_k stated as

$$\Omega_k = \left\{ x \in \Omega \,\middle|\, x_1 \mathbb{E}\{TCP\} + \sum_{k=2}^{K} (x_k + 1) \mathbb{E}\{X_k\} > \tilde{C} \right\}, (3)$$

for $k = 2, ..., K$, and

$$\Omega_1 = \left\{ x \in \Omega \,\middle|\, (x_1 + 1) \mathbb{E}\{TCP\} + \sum_{k=2}^{K} x_k \mathbb{E}\{X_k\} > \tilde{C} \right\}. (4)$$

Here, $\tilde{C}$ (QoS guaranteed capacity in Table I) is the maximum total packet arrival rate to the system packet-level with QoS guarantee. In other words, if the total packet arrival rate is larger than $\tilde{C}$, then the system cannot guarantee the predefined QoS requirement such as the packet loss probability. Since we assume a single cell system, such $\tilde{C}$ is transparent to the underlying system. As to the multi-cell circumstance, the value of $\tilde{C}$ is subject to change due to the inter-cell interference and the number of mobile users in the system.

The packet loss probability and the induced $\tilde{C}$ can be obtained by many existing stochastic loss models. Examples include $M/M/1$ model [26], effective bandwidth model [27], or Gaussian approximation model [15].

B. Call Level Performance: Blocking Probability

Suppose in general that the call arrival process for class k is state-dependent Poisson process with mean $\Lambda_k(x)$ and the service holding time is exponential with mean $1/\Psi_k(x)$ for $x \in \Omega$, indicating that both the arrival process and the service process are state dependent processes. We borrow some notations from [28] and define the $K-$dimensional base vectors as

$$e_0 = \underbrace{(0,0,...,0)}_{K}, \quad e_1 = (1,0,...,0), \quad ... \quad e_K \;=\; (0,0,...,1).$$

and the function $T_i^j(x)$ as $T_i^j(x) = x - e_i + e_j$, i.e., a transition where a single flow from i^{th} class departs and a single flow from j^{th} class arrives to the system. In particular, $T_0^k(x)$ ($T_k^0(x)$) denotes an arrival (a departure) of a flow from class k given that the current state is x.

In particular, we consider the service holding time of a TCP call to be dependent on its transmission rate (i.e., Ψ_1 is state dependent) whereas the holding time of other classes of calls (e.g., voice) are independent of their transmission rate (i.e., $\Psi_k, k = 2,...,K$ are state independent). Assume that the amount of data to be transmitted by TCP call is exponentially distributed with mean ξ.

For any given state $x \in \Omega$, we let the arrive rate equal to the departure rate and obtain the balance equation given by

$$\begin{aligned} &\left[\sum_{k=1}^{K} \Lambda_k 1_{\{T_0^k(x)\in\Omega\}} + \sum_{k=1}^{K} x_k \Psi_k(x) 1_{\{T_k^0(x)\in\Omega\}}\right] P(x) \\ =& \sum_{k=1}^{K} \Lambda_k P(T_k^0(x)) 1_{\{T_k^0(x)\in\Omega\}} \\ +& \sum_{k=1}^{K} (x_k+1)\Psi_k(T_0^k(x)) P(T_0^k(x)) 1_{\{T_0^k(x)\in\Omega\}}. \end{aligned} \tag{5}$$

Next, we can rewrite $\Psi_1(x)$ as follows:

$$\Psi_1(x) \;=\; \begin{cases} \nu_1(x)/\xi & \text{if } x \in \Omega \text{ and } x \notin \Omega_k \;\; \forall k \\ \nu_2(x)/\xi & \text{if } \exists k, \;\; x \in \Omega_k, \end{cases}$$

where $\nu_1(x)$ is the transmission rate of each TCP call (which will be determined by our algorithm) when x is not the boundary state, and $\nu_2(x)$ is the transmission rate of each TCP call when x is one of the boundary states. When the system is not in the boundary state, the system allocates the maximum transmission rate to each TCP call determined by the minimum packet loss probability p_T^l. Otherwise, the system evenly assigns the remaining capacity to each TCP call. Thus, $\nu_1(x)$ and $\nu_2(x)$ can be expressed as

$$\nu_1(x) = \frac{1}{RTT}\sqrt{\frac{3}{2p_T^l}}, \quad \text{and} \quad \nu_2(x) = \frac{\tilde{C} - \sum_{k=2}^{K} x_k \mathbb{E}\{X_k\}}{x_1}.$$

We can obtain the solution to the balance equation (5) similarly as in [28], [29]

$$P(x) = \frac{\Phi(x)}{\Phi(\Omega)}, \qquad x \in \Omega, \tag{6}$$

where $\Phi(x) = \pi(x)\prod_{k=1}^{K} \Lambda_k^{x_k}$ and $\pi(x)$ is calculated recursively by

$$\pi(x) \;=\; \begin{cases} \displaystyle\prod_{k=1}^{K} \frac{1}{\Psi_k^{x_k} x_k!} & \text{if } x \in \Omega \text{ and } x \notin \Omega_k, \;\; \forall k \\ \displaystyle\frac{1}{\tilde{C}} \sum_{k=1}^{K} \pi(T_k^0(x)) & \text{if } \exists k, \;\; x \in \Omega_k, \end{cases}$$

and again, the normalizing factor $\Phi(\Omega)$ is $\Phi(\Omega) = \sum_{x\in\Omega} \Phi(x)$.

Note that the computational complexity will be high for this Erlang type of formula, because when the system size becomes large, the factorial operation in $n!$ results in exponential complexity, i.e., $O((n/e)^n)$. Many techniques have been proposed to reduce the computational overhead as summarized in [30]. Fortunately, the future wireless system tends to be pico-cell or micro-cell, where the capacity of one cell in terms of the number of mobile terminals is around tens and hundred level, rather than thousands or ten thousands.

Remark: We note that the assumption of exponential distribution (i.e., exponential service holding time for non-elastic traffic and exponential amount of data to be transmitted for TCP data) can be relaxed to any arbitrary distribution with the same mean without affecting the steady-state distribution. This is called "insensitivity" property in [28], [29]. Specifically, in [28], [29], the authors give a necessary and sufficient condition for a queueing network in that the network has the insensitivity property if it is 'partially balanced'. (This type of network is called 'Whittle network' in [28], [29].) It turns out that the system model we consider is a special case of Whittle network. Thus, the results in this section can be directly applied to any other distributions with the same mean. In particular, for non-elastic traffic, the exponential service time with mean $1/\Psi_k$ can be extended to any arbitrary distributions with the same mean $1/\Psi_k$, and for elastic traffic, the exponential distribution for the amount of data to be transmitted can be relaxed to any other distributions (See Figures 7 to 11).

VI. Simulation Results

In this section, we investigate the performance of our TCP-aware CAC scheme in Section V via an extensive set of simulations. As in [31], [32], [15], we design and implement an event-driven simulator for multi-class settings and collect results on system dynamics including the call blocking probability, the call-level throughput and the link utilization.

In our simulation, we consider two classes of calls. Note that this is just for simplicity's sake and our scheme can be equally applied to a system with more classes. Class 1 is TCP calls with their transmission rates controlled by the packet loss probability $p(t)$ and Class 2 is voice-type calls with average transmission rate of $E\{\text{voice}\} = 2$ (packets/sec), which corresponds to the voice transmission rate ranging from 2 kbps and 8 kbps by choosing appropriate packet sizes [33]. Here, the value of $p(t)$ is either fixed or dynamically varying according to different scenarios under consideration. The call arrival processes for both classes are characterized by Poisson processes with mean rates λ_1 and λ_2, respectively,

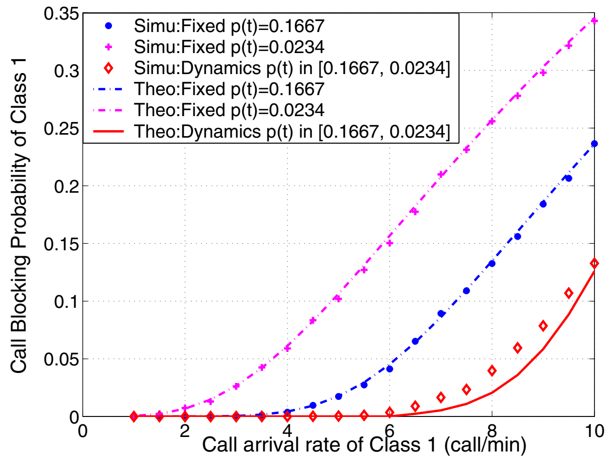

Fig. 2. Blocking probability of class 1 for different TCP AIMD control: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

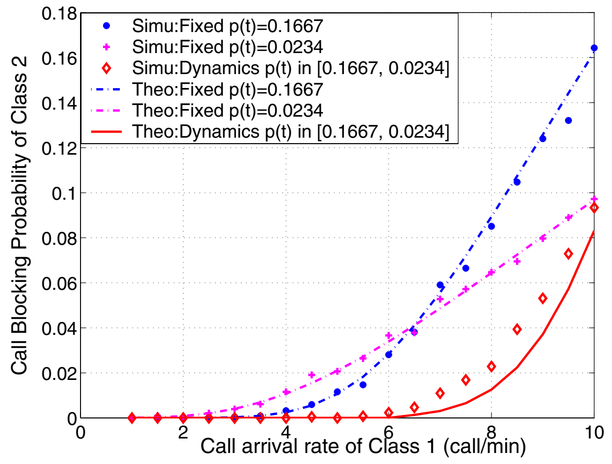

Fig. 3. Blocking probability of class 2 for different TCP AIMD control: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

and the transmission times for two classes are exponentially distributed with mean $1/\mu_1$ and $1/\mu_2$, respectively.

For performance metrics, we consider the call blocking probability, the call-level throughput (calls/min), and the packet-level link utilization. We first look into the case where the amount of data to be transmitted is exponentially distributed, but the service holding time is dependent on the transmission rate. Later on, we also consider different distributions for the amount of data to be transmitted. The packet loss probability of a TCP call ($p(t)$) may change over time and this affects the average throughput of a TCP call (packets/sec) and hence its transmission time as well. In particular, we consider three different strategies for $p(t)$ in our simulation: (i) $p(t) = 0.1667$ is fixed during the transmission for all TCP calls, for which the average TCP throughput becomes $E\{\text{TCP}\} = \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}} = 3$ packets/sec with RTT set to one. This case corresponds to a situation where the system allocates small amount of resource to each TCP call and tries to accommodate more voice-type calls at the expense of longer transmission time for each TCP call. (ii) $p(t) = 0.0234$ and $E\{\text{TCP}\} = \frac{1}{RTT}\sqrt{\frac{3}{2p(t)}} = 8$ packets/sec. This strategy allows higher transmission rate of each TCP call to reduce its the transmission time. (iii) $p(t) \in [0.0234, 0.1667]$ where the system now dynamically changes $p(t)$ following our TCP-aware algorithm as described in Section IV.

Figures 2 and 3 show the call blocking probabilities of Class 1 and Class 2, respectively, as we increase λ_1 (the call arrival rate of Class 1) from 1 to 10 calls/min. In all case, we fix $\lambda_2 = 1$ as our focus is on the system behavior under different amount of Class 1 and 2 calls, not on the exact value of λ_1 and λ_2. The service rates of each class are set to $\mu_1 = 0.5$ and $\mu_2 = 0.5$, respectively. The simulation results are plotted with markers but without lines, and the theoretical results (obtained from (6)) are plotted with lines but without markers. Note that there exists a tradeoff between the case of $p(t) = 0.1667$ and $p(t) = 0.0238$. The blocking probability of Class 1 with $p(t) = 0.1667$ is smaller than that with $p(t) = 0.0238$, while the blocking probability of Class 2 with $p(t) = 0.1667$ becomes larger than that with $p(t) = 0.0238$ when the arrival

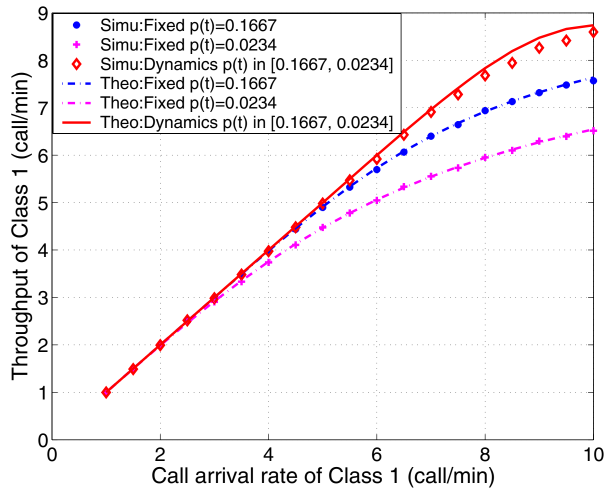

Fig. 4. Throughput of class 1 for different TCP AIMD control: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

rate is larger than 7 (calls/min). This implies that strategies (i) and (ii) have their own advantages and one will outperform the other depending on the settings. However, we note that our proposed scheme with dynamically controlled $p(t)$ yields consistently better performance than both strategies of (i) and (ii).

The same observation holds in terms of the call-level throughput (calls/min) of Class 1 and Class 2 as shown in Figures 4 and 5. Figure 6 shows the packet-level link utilization under the three strategies for $p(t)$. In addition to all better performance at call level, our dynamic $p(t)$ scheme yields the highest link utilization with better packet-level performance.

As mentioned earlier, we also perform extensive simulations with different distributions for the amount of data to be transmitted, but keeping their averages the same. We use log-normal, exponential, and uniform distributions with the same

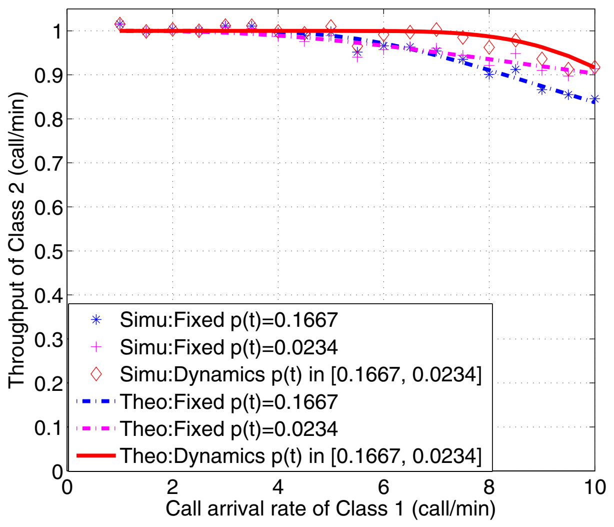

Fig. 5. Throughput of class 2 for different TCP AIMD control: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

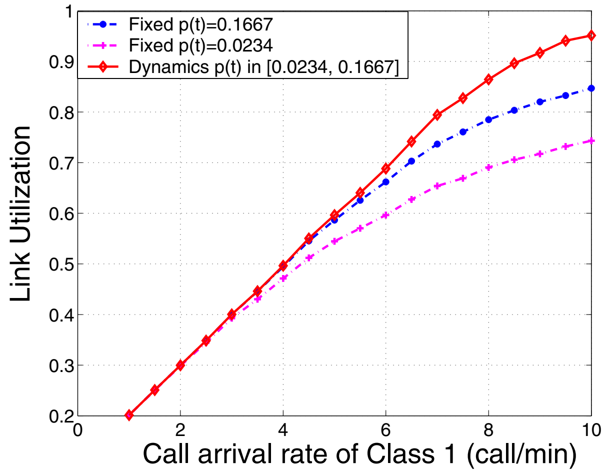

Fig. 6. Link utilization for different TCP AIMD control: (i) fixed $p(t) = 0.1667$; (ii) fixed $p(t) = 0.0238$; (iii) dynamic $p(t) \in [0.0234, 0.1667]$.

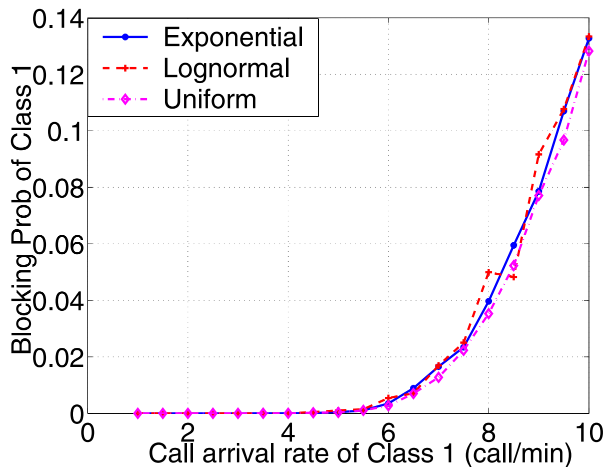

Fig. 7. Blocking probability of class 1 under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution with dynamic $p(t) \in [0.0234, 0.1667]$.

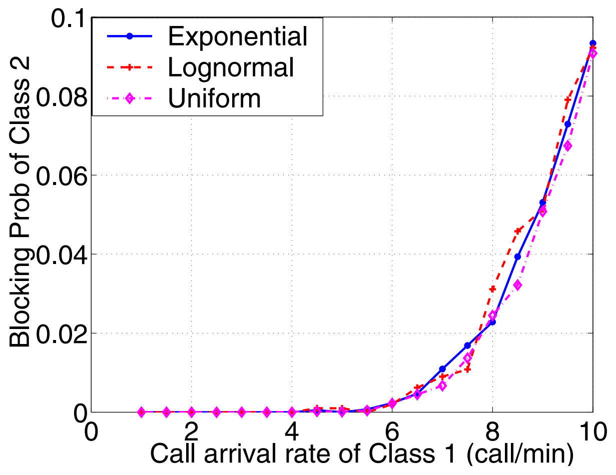

Fig. 8. Blocking probability of class 2 under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution with dynamic $p(t) \in [0.0234, 0.1667]$.

mean.[3] Figures 7 to 11 shows performance metrics (blocking probability, call-level throughput, and link utilization) under the three different distributions, clearly indicating that all the performance metrics remains almost invariant in any case. This asserts that the performance gain of our TCP-aware scheme is largely insensitive to the service distributions and our scheme can be widely applied to more general settings, rather than the exponential distribution only.

VII. Conclusion

Traditional Call Admission Control (CAC) schemes only consider call-level performance and are believed to be sufficient for the circuit-switched wireless network. Since the future wireless network will become packet-switched, the packet-level dynamics cannot be ignored. This is especially true when the TCP-type elastic applications are running over such packet-switched wireless networks, because TCP congestion algorithm will exhaust all the available bandwidth until packet loss occurs. In order to effectively utilize the elastic feature of TCP applications, we have proposed a TCP-AIMD aware CAC scheme based on a two-level framework that takes both the call-level and the packet-level dynamics directly into account. The scheme is general enough and can be tailored to cope with specific packet switched wireless networks with proper modifications. Our extensive simulation results show that the proposed scheme can significantly improve the system performance from CAC perspective such as the call blocking probability (call-level metric), throughput (calls/min, call-level metric), and the link utilization (packet-level metric), which are in good agreement with our analysis.

[3]We have also tested other various distributions and observed the same performance. Due to space constraint, however, we do not include them here.

References

[1] S. Ravot, "TCP transfers over high latency bandwidth networks grid DT," in *Proc. The First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003)*, Feb 2003.

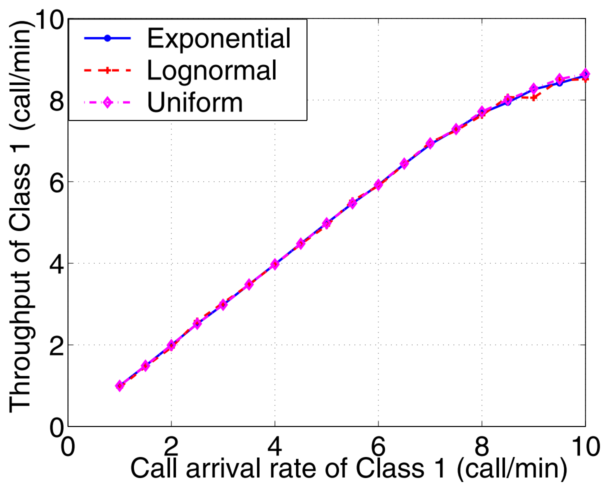

Fig. 9. Throughput of class 1 under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution. Dynamic $p(t) \in [0.0234, 0.1667]$.

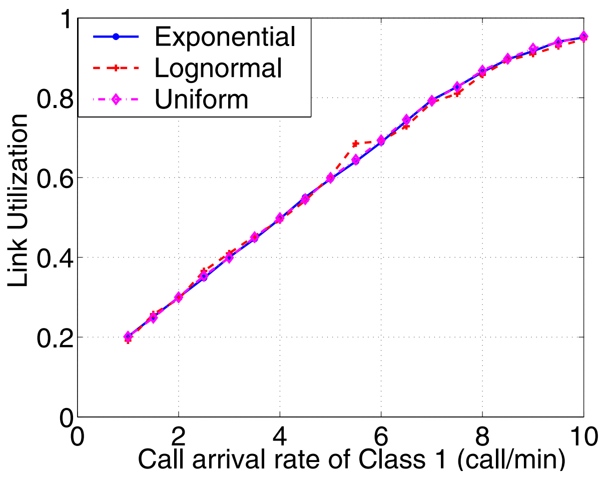

Fig. 11. Link utilization under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution with dynamic $p(t) \in [0.0234, 0.1667]$.

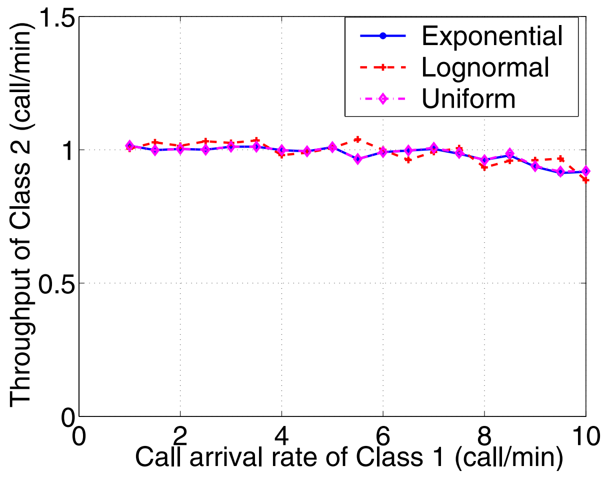

Fig. 10. Throughput of class 2 under different distributions of workload: Exponential distribution; Lognormal distribution; Uniform distribution with dynamic $p(t) \in [0.0234, 0.1667]$.

[2] M. Jain, R. Prasad, and C. Dovrolis, “Socket buffer auto-sizing for maximum TCP throughput,” in *J. Grid Comput.*, vol. 1, pp. 361–376, 2004.

[3] C. Na and T. S. Rappaport, “Measured wireless LAN public hotspot traffic statistics,” *Electron. Lett.*, vol. 40, pp. 1202–1203, Sept 2004.

[4] M. C. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” in *Proc. ACM MOBICOM 2002*.

[5] M. C. Chan and R. Ramjee, “Improving TCP/IP performance over third generation wireless networks,” in *Proc. IEEE INFOCOM 2004*.

[6] J. Liu and S. Singh, “ATCP: TCP for mobile ad-hoc networks,” *IEEE J. Sel. Areas Commun.*, vol. 19, pp. 1300–1315, July 2001.

[7] K. Brown and S. Singh, “M-TCP: TCP for mobile celluar networks,” *ACM Computer Commun. Rev.*, vol. 27, no. 5, pp. 19–43, 1997.

[8] A. Bakre and B. R. Badrinath, “I-TCP: indirect TCP for mobile hosts,” in *Proc. 15th International Conference on Distributed Computing Systems 1995*.

[9] S. Y. Hui and K. Yeung, “Challenges in the migration to 4G mobile systems,” *IEEE Commun. Mag.*, pp. 54–59, Dec. 2003.

[10] H. Honkasalo, K. Pehkonen, M. T. Niemi, and A. T. Leino, “WCDMA and WLAN for 3G and beyond,” *IEEE Wireless Commun. Mag.*, pp. 14–18, April 2002.

[11] W. Wang, X. Wang, and A. Nilsson, “Energy-efficient bandwidth allocation in wireless networks: algorithms, analysis, and simulation,” *IEEE Trans. Wireless Commun.*, vol. 5, pp. 1103–1114, April 2006.

[12] R. G. Akl, M. V. Hegde, and M. Naraghi-Pour, “Mobility-based CAC algorithm for arbitrary call-arrival rates in CDMA celluar systems,” *IEEE Trans. Veh. Technol.*, vol. 54, pp. 639–651, March 2005.

[13] C. W. Leong, W. Zhuang, Y. Cheng, and L. Wang, “Call admission control for integrated on/off voice and best-effort data services in mobile celluar communications,” *IEEE Trans. Commun.*, vol. 52, pp. 778–790, May 2004.

[14] Y. Fang and Y. Zhang, “Call admission control scheme and performance analysis in wireless mobile networks,” *IEEE Trans. Veh. Technol.*, vol. 51, pp. 371–382, March 2002.

[15] M. Ghaderi and R. Boutaba, “Call admission control for voice/data integration in broadband wireless networks,” *IEEE Trans. Mobile Comput.*, vol. 5, pp. 193–207, March 2006.

[16] D. Zhang, X. Shen, and J. W. Mark, “Radio resource management for cellular CDMA systems supporting heterogeneous services,” *IEEE Trans. Mobile Comput.*, vol. 11, pp. 32–39, April-June 2003.

[17] S. B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J. W. Roberts, “Statistical bandwidth sharing: a study of congestion at flow level,” in *Proc. ACM SIGCOMM’01*, Aug. 2001.

[18] S. Borst, “User-level performance of channel-aware scheduling algorithms in wireless data networks,” in *Proc. IEEE INFOCOM*, July 2003.

[19] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, “CDMA/HDR: a bandwidth-efficient high-speed wireless data service for nomadic users,” *IEEE Commun. Mag.*, pp. 70–77, July 2000.

[20] S. Floyd, “TCP and explicit congestion notification,” *ACM Computer Commun. Rev.*, vol. 24, pp. 10–23, Oct. 1994.

[21] V. Jacobson, “Congestion avoidance and control,” *ACM Computer Commun. Rev.*, vol. 8, pp. 314–329, Aug. 1988.

[22] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, “Modeling TCP Reno performance: a simple model and its empirical validation,” *IEEE Trans. Netw.*, vol. 8, pp. 133–145, April 2000.

[23] M. Mathis, J. Semke, and J. Mahdavi, “The macroscopic behavior of the TCP congestion avoidance algorithm,” in *ACM Sigcomm 1997*, vol. 27, July 1997.

[24] M. H. Ahmed, “Call admission control in wireless networks: a comprehensive survey,” *IEEE Commun. Surveys & Tutorials*, vol. 7, no. 1, pp. 49–68, 2005.

[25] B. Li, L. Li, B. Li, K. M. Sivalingam, and X. Cao, “Call admission control for voice/data integrated cellular networks: performance analysis and comparative study,” *IEEE J. Sel. Areas Commun.*, vol. 22, pp. 706–718, May 2004.

[26] L. Kleinrock, *Queueing Systems, Volume I: Theory*. Wiley Interscience, 1975.

[27] E. W. Knightly, “H-BIND: a new approach to providing statistical performance guarantees to VBR traffic,” in *Proc. IEEE INFOCOM’96*, March 1996.

[28] R. F. Serfozo, *Introduction to Stochastic Networks*. Springer, 1999.

[29] T. Bonald and A. Proutiere, “Insensitivity in processor-sharing networks,” *Performance Evaluation*, vol. 49, pp. 193–209, Oct. 2002.

[30] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer-Verlag, London, 1995.

[31] K. Nahm, A. Helmy, and C. C. J. Kuo, "TCP over multihop 802.11 networks: issues and performance enhancement," in *Proc. ACM MobiHoc'05*, May 2005.

[32] S. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proc. ACM MobiHoc'05*, May 2005.

[33] "UMTS," available at http://www.3GPP.org.

Xinbing Wang received the B.S. degree (with honors) from the Department of Automation, Shanghai Jiaotong University, Shanghai, China, in 1998, and the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001. He received his Ph.D. degree major in the Department of Electrical and Computer Engineering, minor in Department of Mathematics, North Carolina State University, Raleigh, NC, USA in 2006. He is now a lecture with Department of Electronic Engineering, Shanghai Jiaotong University, China. His research interests include Resource Allocation and Management in Mobile and Wireless Networks; TCP Asymptotics Analysis; Cross Layer Call Admission Control; Asymptotics Analysis of Cooperative Wireless Networks, and Congestion Control over Wireless Ad-hoc and Sensor Networks. He is a member of Technical Program Committee of IEEE ICC 2007, IEEE WCNC 2007, IEEE ICCCN 2007 and IEEE IPCCC 2007.

Do Young Eun received his B.S. and M.S. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1995 and 1997, respectively, and Ph.D. degree from Purdue University, West Lafayette, IN, in 2003. Since August 2003, he has been an Assistant Professor with the Department of Electrical and Computer Engineering at North Carolina State University, Raleigh, NC. His research interests include network modeling and analysis, congestion control, resource allocation, and ad-hoc/sensor networks. He is a member of Technical Program Committee of IEEE INFOCOM 2005, 2006, IEEE ICC 2005, 2006, IEEE Globecom 2005, and IEEE IPCCC 2006. He received the Best Paper Award in the IEEE ICCCN 2005 and the NSF CAREER Award 2006.

Wenye Wang (M'98/ACM'99) received the B.S. and M.S. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 1986 and 1991, respectively. She also received the M.S.E.E. and Ph.D. degree from Georgia Institute of Technology,Atlanta, Georgia in 1999 and 2002, respectively. She is now an Assistant Professor with the Department of Electrical and Computer Engineering, North Carolina State University. Her research interests are in mobile and secure computing, quality-of-service (QoS) sensitive networking protocols in single- and multi-hop networks. She has served on program committees for IEEE INFOCOM, ICC, ICCCN in 2004. Dr. Wang is a recipient of NSF CAREER Award in 2006. She has been a member of the Association for Computing Machinery since 2002.