MAQ: A Multiple Model Predictive Congestion Control Scheme for Cognitive Radio Networks

Kefan Xiao, *Student Member, IEEE*, Shiwen Mao, *Senior Member, IEEE*, and Jitendra K. Tugnait, *Fellow, IEEE*

***Abstract*— In this paper, we investigate the problem of robust congestion control in infrastructure-based cognitive radio networks (CRN). We develop an active queue management algorithm, termed MAQ, which is based on multiple model predictive control. The goal is to stabilize the TCP queue at the base station under disturbances from the time-varying service capacity for secondary users. The proposed MAQ scheme is validated with extensive simulation studies under various types of background traffic and system/network configurations. It outperforms two benchmark schemes with considerable gains in all the scenarios considered.**

***Index Terms*— Active queue management, cognitive radio network, congestion control, multiple model predictive control, stability.**

I. Introduction

THE wide availability of mobile devices and services has triggered unprecedented demand for wireless capacity, and the demand is predicted to increase drastically in the near future. In parallel with the effort on exploiting new spectrum [1], [2] and new antenna techniques [3], [4], cognitive radio network (CRN) has been recognized as a promising solution for more capacity by exploiting underutilized licensed spectrum [5]. In CRNs, licensed users (or, Primary User (PU)) share spectrum with CRN users (or, Secondary User (SU)). With the dynamic spectrum access (DSA) approach, SUs detect and access unused licensed channels, where the tension between PU protection and SU capacity gain should be carefully balanced [6]. In particular, the SUs have a lower priority for channel access and their service capacity usually varies over time as affected by the PU transmission behavior.

Although CRN has been well-studied in the past decade, the mainstream research effort has been focused on spectrum sensing, access, leasing, and policy related aspects. Some other important network-level problems, such as congestion control, have not been well studied [7]. Since most network applications (e.g., even video streaming from many commercial video-sharing websites) are based on TCP, supporting TCP is critical for enabling such applications in CRNs, which is also crucial for the success of CRNs [8]. However, this problem has only been investigated in a few prior works [9]–[11]. In [9] and [10], the TCP congestion control mechanism is modified for the CRN environment, while in [11], a partially observable Markov decision process (POMDP) based approach is proposed to maximize the SU throughput by jointly adjusting spectrum sensing, access, and the physical-layer modulation and coding scheme. Although interesting results are derived, the existing schemes are either offline [11] or based on heuristics [9], [10].

Motivated by these interesting works, we investigate the problem of robust congestion control in CRNs. It is well-known that TCP does not work well in wireless networks, largely due to the fact that it does not distinguish between packet loss due to congestion or transmission errors. Many effective solutions have been proposed in the literature, such as split-TCP or making the wireless link more reliable [12]. In a general wireless network, capacity variation is mainly caused by fading, shadowing, and interference. In CRNs, the additional challenge is the impact of PU transmissions on SU TCP sessions, which may even have a rate of zero during sensing periods when all the SUs stop transmission to sense the channels. During the transmission period, the capacity that is available for SU TCP sessions also varies over time due to PU transmissions, and the variation could be large and occur at various timescales from session to packet levels. These pose considerable disturbance to the TCP feedback control mechanism. There is a critical need for robust congestion control mechanisms in such a highly dynamic network environment.

With an essential understanding of the problem, i.e., voluminous variation of capacity and uncertainty introduced by PU activity and the wireless environment, we proposed an algorithm based on the multiple model predictive control (MMPC) framework, for its robustness over multiple disturbances and multiple system models. The traditional model predictive control (MPC) utilizes a model to predict system output in response to input signals propagating into the future, while taking both optimal calculation and control action into consideration. However, MPC normally considers only one model and a single disturbance, which makes it not effective for our problem. The MMPC, on the other hand, considers multiple system models and disturbances with a disturbance estimation and rejection mechanism, and thus can enhance the robustness of the system. At each step, an optimization problem is formulated by taking future p steps into consideration,

Manuscript received September 25, 2016; revised January 12, 2017; accepted February 10, 2017. Date of publication March 13, 2017; date of current version April 7, 2017. This work was supported in part by the U.S. NSF under Grant CNS-1320664, and in part by the Wireless Engineering Research and Education Center at Auburn University. This work was presented in part at the IEEE GLOBECOM, Washington, DC, USA, 2016. The associate editor coordinating the review of this paper and approving it for publication was Q. Li.

The authors are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201 USA (e-mail: kzx0002@tigermail.auburn.edu; smao@ieee.org; tugnajk@eng.auburn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2017.2669322

in which p is the prediction horizon. And the decision results are m steps of control moves, where m is the control horizon. Generally, the design process should be conservative about the accuracy of the estimation of future states. Therefore, we have $m < p$ in most cases.

In this paper, we consider an infrastructure-based CRN, where multiple SUs maintain TCP connections with various TCP servers in the Internet. The TCP sessions share a buffer at the CRN base station (BS) and the last hop, and bottleneck, of the TCP connections is the downlink of the CRN. The BS advertises a zero-size receive window on behalf of the SUs to inform the TCP senders of the sensing period and to freeze the TCP servers during the sensing period [9]. Since the classic additive-increase-multiple-decrease (AIMD) algorithm incorporated in TCP does not deal well with the frozen period and capacity variations, we aim to develop a new congestion control mechanism that is robust to such disturbances.

Specifically, we exploit active queue management (AQM) to deal with the CRN congestion control problem. AQM is a class of packet dropping/marking mechanisms implemented at the router queue to support end-to-end congestion control. We develop a robust AQM mechanism to stabilize the queue length at the CRN BS, which can not only yield a relative stable queueing delay, but also absorb the disturbances caused by busty background traffic or capacity variation. The proposed scheme, termed MAQ, is based on multiple model predictive control (MMPC) that integrates the estimation and prediction of multiple models with different weights, which can significantly enhance the robustness of the controller to reject disturbances from the environment [13]. We evaluate the performance of the proposed scheme with extensive NS2 simulations, such as under responsive and non-responsive background traffic, varying number of TCP connections, various propagation delays, and various reference queue lengths. The simulation study shows that MAQ can effectively stabilize the TCP buffer in all the scenarios we simulated, and outperforms two benchmark schemes with considerable gains. In summary, our contribution includes the following three aspects.

- We identify the main challenges of congestion control in CRNs. We modify the flow model accordingly and propose a TCP-compatible method to deal with the sensing period in CRN.
- We propose an algorithm based on MMPC for effective congestion control in CRNs. In the multiple models bank building process, we show how to choose operation points and present detailed parameter tuning and computation complexity analysis.
- Extensive simulations are conducted based on NS2. We test various scenarios under various traffic patterns. We find that the proposed algorithm can converge to the set point and is superior over two benchmark algorithms on both robustness and quality of service.

The remainder of this paper is organized as follows. We present the system model and problem formulation in Section II. We discuss the MAQ design in Section III and validate its performance in Section IV. Related work is reviewed in Section V and Section VI concludes this paper.

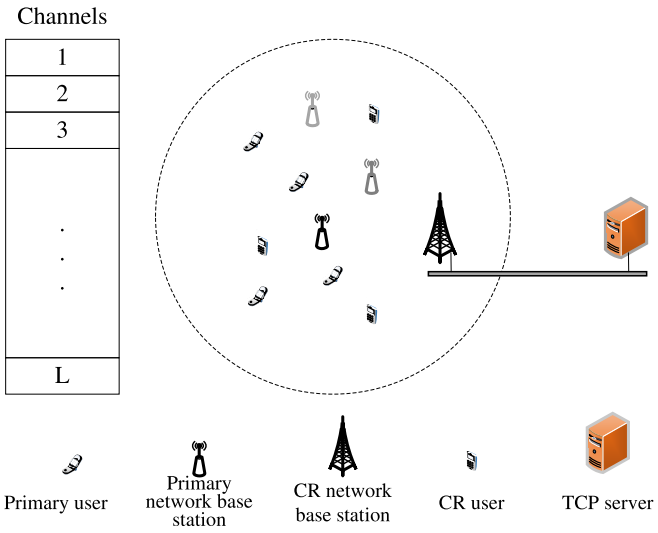


Fig. 1. Network model of CR network.

II. System Model and Problem Formulation

A. Network Model

We consider a primary network with M licensed channels. The primary users transmit on the licensed channels from time to time, and the availability of each channel (i.e., transmission opportunities for SUs) follows a certain on-off process [6]. There is a CRN collocated with the primary network, consisting of a CRN base station (BS) and N active SUs. The CRN BS and SUs cooperatively sense the licensed channels to identify transmission opportunities for SUs. Since spectrum sensing is a well-studied problem, we assume that an effective spectrum sensing scheme is in force and the sensing results are accurate with a high probability. The infrastructure-based CRN network model is illustrated in Fig. 1.

In the CRN, each active SU maintains a TCP connection with a TCP server in the Internet cloud (e.g., downloading a large file). We assume the channel bonding/aggregation technique is used, such that the BS and SUs can transmit and receive on multiple assigned channels simultaneously to make use of all the available spectrum [14]. Let the aggregated usable capacity be $C(t)$, which is time-varying as availability of the licensed channels vary over time. All the TCP connections share the downlink capacity of the BS with time division multiplexing (TDM), assuming negligible uplink feedback. Due to the mismatch of capacity between wired and wireless links, it is reasonable to assume that the CRN BS is the bottleneck of the TCP connections.

We aim to develop a congestion control mechanism to stabilize the bottleneck queue at the CRN BS for the TCP connections under time-varying capacity $C(t)$. In addition to the time-varying capacity $C(t)$, the sensing period of the CRN also poses a challenge for congestion control, during which $C(t) = 0$ since the BS and all the SUs stop transmission to sense the licensed channels. We follow the approach in [9] to freeze the TCP senders during the sensing period, assuming that the TCP servers obtain the sensing schedule of the CRN during the session setup phase and the zero-size receive

window mechanism is used [9]. Thus the sensing period only has a negligible impact on congestion control except for a fixed small cost to the overall throughput.

B. Fluid Flow Model for TCP Sessions

Ignoring the sensing period of the CRN, during the transmission period, the dynamics of the multiple-TCP system can be modeled by a fluid flow model [15]. Different from many existing AQM algorithms, we adopt the *drop-from-front* mechanism [16], i.e., when there is incipient congestion, the head-of-line (HOL) packet in the bottleneck queue will be dropped (instead of the packet at the tail). This way, the TCP server can infer congestion more quickly (i.e., without the queueing delay) and react faster to congestion as well as the changing capacity in the CRN.

The dynamics of the TCP connections sharing a queue at the CRN BS can be derived by slightly modifying the model in [15] as follows.

$$\dot{W}(t) = \frac{1}{R(t)} - \beta W(t)\frac{W(t-T_p)}{R(t-T_p)}p(t-T_p) \tag{1}$$

$$\dot{q}(t) = \begin{cases} N(t)\frac{W(t)}{R(t)} - C(t), & q > 0 \\ \max\left\{0, N(t)\frac{W(t)}{R(t)} - C(t)\right\}, & q = 0, \end{cases} \tag{2}$$

where $\dot{f}(t)$ donates the time-derivative of a function $f(t)$. The other functions are defined as follows.

- $q(t)$: the queue length at the CRN BS and $q(t) \in [0, B]$, where B is the maximum buffer size.
- $C(t)$: the downlink capacity of the CRN BS in packets/s.
- $R(t)$: the round trip time (RTT) of a TCP connection.
- $p(t)$: the packet drop probability of AQM.
- T_p: the propagation delay of a TCP connection, i.e., RTT minus queueing delay at the CRN BS.
- $N(t)$: the number of active TCP connections going through the CRN BS.
- $W(t)$: the congestion window size of the TCP source.
- β: the window multiplicative decrease parameter, which a constant and is usually set to $1/2$ (sometimes as $\ln(2)$) [17].

This model is based on the fluid flow traffic model and is a system of delayed stochastic differential equations. It ignores the TCP time out mechanism to make the problem manageable. Since drop-from-front is adopted, the drop probability p is actually delayed by a propagation delay T_p to take effect at the TCP server.

C. Linearization and Discretization

1) Linearization: We first build the model bank for the proposed MMPC scheme by choosing *multiple* operating points (each corresponding to a model) and linearizing the system around them. It's obvious that the more models, the more accurate the representation will be. However, the computational complexity will also be higher. For practical systems, usually three to five models should be sufficient. We choose π different reference capacity values denoted as C_i, $i = 1, 2, \ldots, \pi$, in descending order as follows.

- When $\pi = 3$: $\bar{C}$, $\bar{C} \pm \sigma_C$;
- When $\pi = 5$: $\bar{C}$, $\bar{C} \pm 0.75 \times \sigma_C$, $\bar{C} \pm 1.5 \times \sigma_C$,

where $\bar{C} = \mathrm{E}[C(t)]$ is the average link capacity and σ_C is the standard deviation of $C(t)$, which can be obtained from historical data.

A change in the capacity will cause immediate change of the queue length. The magnitude depends on how quickly the TCP source can react to the changing capacity. Given an expected queuing delay D_q and considering the chosen reference capacities, we choose the corresponding π reference queue lengths as q_i, $i = 1, 2, \ldots, \pi$, in ascending order as follows.

- When $\pi = 3$: $q_1 = \bar{C}D_q$, $q_1 \pm \sigma_C D_q$;
- When $\pi = 5$: $q_1 = \bar{C}D_q$, $q_1 \pm 0.75 \times \sigma_C D_q$, $q_1 \pm 1.5 \times \sigma_C D_q$.

The operating point i of the system, denoted as five-tuples $(W_i, q_i, p_i, C_i, R_i)$, can be solved from the system equations (1) and (2) by setting $\dot{W} = 0$ and $\dot{q} = 0$, for $i = 1, 2, \ldots, \pi$. Assuming the system is stabilized at operating point i, we have $W(t) = W(t - T_p)$ and $R(t) = R(t - T_p)$. It follows that

$$\beta W_i^2 p_i = 1; \quad W_i = \frac{R_i C_i}{N}; \quad R_i = T_p + \frac{q_i}{C_i}. \tag{3}$$

Denote the deviations from operating point i as $\delta W_i = W(t) - W_i$, $\delta q_i = q(t) - q_i$, $\delta p_i = p(t) - p_i$, and $\delta C_i = C(t) - C_i$. Further denote model i state as $x_i = [\delta q_i, \delta W_i]^T$ and output as $y_i = \delta q_i$, and define $\delta p_i(t - T_p)$ as $u_i(t - T_p)$. The linearized system around operating point i can be obtained as follows.

$$\dot{x_i}(t) = A_i x_i(t) + B_i u_i(t - T_p) + D_i \delta C_i(t) \tag{4}$$

$$y_i(t) = Q_i x_i(t) + \upsilon(t). \tag{5}$$

Equation (4) is the state equation and (5) is the output equation, where $\upsilon(t)$ accounts for the disturbance from the uncertainty nature of control input $u_i(t - T_p)$. The coefficient matrices are given below. The details are omitted for brevity.

$$A_i = \begin{bmatrix} -\frac{1}{R_i} & \frac{N}{R_i} \\ 0 & -R_i^2 \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ -\frac{\beta R_i C_i^2}{N^2} \end{bmatrix},$$

$$D_i = \begin{bmatrix} -\frac{T_p}{R_i} & 0 \end{bmatrix}^T, \quad Q_i = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

2) Discretization: Since packets arriving and departing in a discrete manner, the entire system can be viewed as sampling from the continuous-time model with a finite sampling rate. Denote the sampling period as T_s, which is a constant and is sufficiently small, such that every propagation delay T_p is an integer multiple of T_s as $T_p = n_0 T_s$.

Let $x[k]$ represent the kth sample $x(kT_s)$. Then the ith linear model (around operating point i) can be discretized as

$$x_i[k+1] = \Phi_i x_i[k] + \Gamma_i u_i[k - n_0] + \Delta_i \delta C_i(k) \tag{6}$$

$$y_i[k] = Q_i x_i[k] + \upsilon_i[k]. \tag{7}$$

The new coefficients Φ_i, Γ_i, and Δ_i are derived with the standard discretization algorithm and are given

below.

$$\Phi_i = e^{A_i T_s} = \begin{bmatrix} e^{-\frac{T_s}{R_i}} & \frac{C_i N R_i}{C_i R_i - 2N}\left(e^{-\frac{2NT_s}{R_i^2 C_i}} - e^{-\frac{T_s}{R_i}}\right) \\ 0 & e^{-\frac{2NT_s}{R_i^2 C_i}} \end{bmatrix},$$

$$\Gamma_i = \left(\int_0^{T_s} e^{A_i \tau} d\tau\right) B = \beta \frac{C_i^3 R_i^3}{N} \times \begin{bmatrix} \frac{C_i R_i}{2N(C_i R_i - 2N)} e^{-\frac{2NT_s}{R_i^2 C_i}} - \frac{1}{C_i R_i - 2N} e^{-\frac{T_s}{R_i}} - \frac{1}{2N} \\ \frac{1}{2N^2}\left(e^{-\frac{2NT_s}{R_i^2 C_i}} - 1\right) \end{bmatrix},$$

$$\Delta_i = \left(\int_0^{T_s} e^{A_i \tau} d\tau\right) D_i = \begin{bmatrix} T_p\left(e^{-\frac{T_s}{R_i}} - 1\right) \\ 0 \end{bmatrix}.$$

The above derivation assumes $C_0 R_i - 2N \neq 0$. Otherwise, when $C_0 R_i = 2N$, the coefficients become

$$\Phi_i = \begin{bmatrix} e^{-\frac{T_s}{R_i}} & \frac{NT_s}{R_i}\left(e^{-\frac{T_s}{R_i}}\right) \\ 0 & e^{\frac{T_s}{R_i}} \end{bmatrix}$$

$$\Gamma_i = 4\beta N \begin{bmatrix} -\left(1 + \frac{T_s}{R_i}\right) e^{-\frac{T_s}{R_i}} - 1 \\ e^{-\frac{T_s}{R_i}} - 1 \end{bmatrix}.$$

We also consider $\delta C[k]$, the variation of link capacity, as a state variable, which accounts for both capacity variations and other disturbances. $\delta C[k]$ evolves as follows.

$$\delta C[k+1] = \delta C[k] + \omega[k], \tag{8}$$

where $\omega[\cdot]$ is the external disturbance. Due to the uncertainty such as modeling error or unknown disturbance, it is usually hard, if not impossible, to drive the system to converge to the operating point without an offset. In order to ensure convergence, augmenting the system state with the disturbance is a feasible approach, since it would be possible to estimate the disturbance and then suppress its impact. The augmented state is $x_i^a[k] = [\delta q_i, \delta W_i, \delta C_i]^T$. Finally, the discretized system can be rewritten as

$$x_i^a[k+1] = \Phi_i^a x_i^a[k] + \Gamma_i^a u[k - n_0] + \Theta_i^a w[k] \tag{9}$$

$$y_i[k] = Q_i^a x_i^a[k] + v_i[k], \tag{10}$$

where the coefficient matrices are

$$\Phi_i^a = \begin{bmatrix} \Phi_i & \Delta_i \\ 0 & I \end{bmatrix}, \quad \Gamma_i^a = \begin{bmatrix} \Gamma_i \\ 0 \end{bmatrix}, \quad \Theta_i^a = \begin{bmatrix} 0 \\ I \end{bmatrix}, \quad Q_i^a = [Q_i, 0].$$

III. MAQ Design

In the previous section, we obtain multiple models for the CRN congestion control system by linearization at several operating points, discretization, and state augmentation. Although it is possible to design one controller for each model respectively, it is hard to determine which controller should be active. Instead, we use the entire set of models as a *model bank* to estimate the system state and output. The estimation part is crucial in our control algorithm design. Due to the propagation delay, the control output, i.e., the dropping probability p, will take effect at the TCP server after the propagation delay T_p.

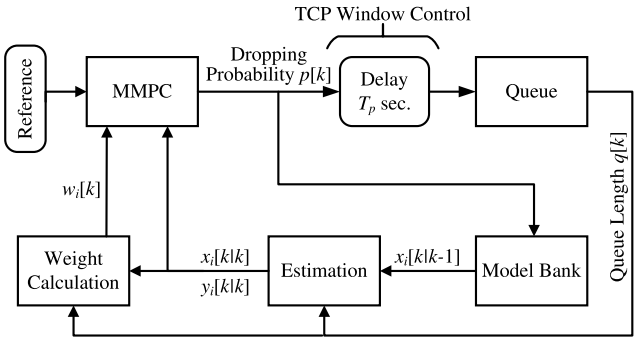


Fig. 2. Control system structure.

The entire control system is shown in Fig. 2. When there is incipient congestion, packets are dropped from front with a certain probability at the CRN BS. It's worth noting that we calculate dropping probability $p[k]$ based on the measurement $q[k]$ at time step k, as shown in Fig. 2. However, the probability $p[k]$ will take effect on the queue length after a propagation delay. For the control loop, we have discussed the model bank in the previous section but we still need to briefly restate it in the control context. We present the design of the MAQ scheme in rest of this section.

A. Estimation

In the model bank we developed, the noise is usually not known in advance in practice. It is necessary to utilize the previous information to estimate both the augmented state and output. Let $q[k]$ be the measured queue length at step k. We use Kalman Filter to perform state and output estimation as follows.

$$\hat{x}_i^a[k|k-1] = \Phi_i^a \hat{x}_i^a[k-1|k-1] + \Gamma_i^a u_i[k - n_0 - 1] \tag{11}$$

$$\hat{x}_i^a[k|k] = \hat{x}_i^a[k|k-1] + L_i[k](q[k] - q_i - Q_i^a \hat{x}_i^a[k|k-1]) \tag{12}$$

$$\hat{y}_i^a[k|k] = Q_i^a \hat{x}_i^a[k|k], \tag{13}$$

where $\hat{x}_i^a[k|k-1]$ is the estimation of step k based on step $(k-1)$ information, $\hat{x}_i^a[k|k]$ represents the updated state at step k based on the measurement $q[k]$ at time step k, and $L_i[k]$ is the Kalman gain of model i. The "hat" notation indicates that they are estimations. Eq. (13) is the output estimation. The Kalman gain $L_i[k]$ can be derived from the following process.

$$L_i[k] = P_i[k](Q_i^a)^T (Q_i^a P_i[k-1](Q_i^a)^T + G_i)^{-1} \tag{14}$$

$$P_i[k] = \Phi_i^a P_i[k-1](\Phi_i^a)^T + \Theta_i^a O_i (\Theta_i^a)^T - \Phi_i^a P_i[k-1](Q_i^a)^T \times (Q_i^a P_i[k-1](Q_i^a)^T + G_i)^{-1} Q_i^a P_i[k-1](\Phi_i^a)^T, \tag{15}$$

where $P_i[k]$ is the estimation error covariance of model i at step k.

This algorithm works in an iterative manner. Parameters O_i and G_i in (15) are stochastic terms that account for the variance of state interference and output, respectively. Since usually the variance is unknown, O_i and G_i are the parameters to be tuned. In the MAQ design, both O_i and G_i are scalar and are tuned for each of the models.

B. Weight Calculation

Given the estimations of the state and output of each model, a more accurate approximation to the real system dynamics can be obtained by probabilistically integrating the models. We first derive the weight for each model based on their *residual* at time step k, denoted as $\xi_i[k]$, which is the difference between the real measurement and estimation output of each model. Noticing that the measurement $q[k]$ is the queue length while the estimation $\hat{y}_i[k|k]$ is the deviation from the operating point, we have

$$\xi_i[k] = q[k] - q_i - \hat{y}_i[k|k]. \tag{16}$$

Under the general assumption that the weights are Gaussian, the probability that model i represents the system at step k can be derived with the Bayesian theorem as

$$\rho_i[k] = \frac{\exp(-0.5\lambda\xi_i^2[k])\rho_i[k-1]}{\sum_{j=1}^{\pi}\exp(-0.5\lambda\xi_j^2[k])\rho_j[k-1]}. \tag{17}$$

In (17), the exponential term is due to the Gaussian distribution of the weights. It would be beneficial that the rejecting speed of the unsuitable models is exponential. The coefficient λ determines the sensitivity to the residual. Furthermore, the formula is recursive; once $\rho_i[k]$ is 0 at some step k, it will remain at 0 even if the model is more accurate than others. We thus set a lower limit $\mu > 0$, such that if $\rho_i[k]$ is lower than μ, it will be set to μ.

The weights are then calculated by normalizing the probabilities, as

$$w_i[k] = \begin{cases} \frac{\rho_i[k]}{\sum_{j=1}^{\pi}\rho_j[k]}, & \rho_i[k] > \mu \\ 0, & \rho_i[k] \le \mu. \end{cases} \tag{18}$$

In the initialization phase, all the weights are set to $1/\pi$ since there is no a priori information about the system.

C. MMPC Control Law

We are now ready to derive the MMPC control law. Due to the propagation delay, the control signal $u_i[k]$ affects the queue length at time $(k+n_0)$. It is thus necessary to estimate the n_0-step ahead state and output, i.e., at time $(k+n_0)$. We estimate state $x_i^a[k+n_0]$ as

$$\begin{aligned}\hat{x}_i^a[k+n_0|k] &= \Phi_i^a\hat{x}_i^a[k+n_0-1|k] + \Gamma_i^a u_i[k-1] \\ &= (\Phi_i^a)^{n_0}\hat{x}[k|k] + \sum_{j=1}^{n_0}(\Phi_i^a)^{j-1}\Gamma_i^a u_i[k-j]. \end{aligned}\tag{19}$$

The output of each model at time $(k+n_0)$ can also be estimated. A more accurate estimation, $\bar{y}$, can be obtained as the weighted average of the estimated outputs. It follows that

$$\bar{y}[k+n_0|k] = \textstyle\sum_{i=1}^{\pi} w_i[k]\hat{y}_i[k+n_0|k], \tag{20}$$

where $\hat{y}_i[k+n_0|k]$ is derived with the estimation precess. Recall that output $\hat{y}_i[k+n_0|k]$ is the difference between the estimated queue length and the ith operating point q_i. Substituting $\hat{y}_i[k+n_0|k] = \hat{q}[k+n_0] - q_i$ into (20), we have

$$\bar{y}[k+n_0|k] = \hat{q}[k+n_0] - \sum_{i=1}^{\pi} w_i[k]q_i. \tag{21}$$

Furthermore, the control signal $u_i[k]$ is the difference between the dropping probability and its operating point value, i.e., $u_i[k] = p[k] - p_i$, while $p[k]$ is to be derived. The control objective function is formed with a prediction horizon of s time steps and a control horizon of m time steps. As discussed, the prediction horizon starts from time step $(k+n_0)$. By solving the optimization problem with this objective function at every time step, a series of optimal control signals $\{\Delta p[k],\ \Delta p[k+1],\ \ldots,\ \Delta p[k+s]\}$ can be derived, where $\Delta p[k] = p[k+1] - p[k]$. The first control signal is then chosen as our result. The $(k+n_0+j)$th step prediction is the weighted average of all the model outputs at the same time step, i.e.,

$$\bar{y}[k+n_0+j|k] = \sum_{i=1}^{\pi} w_i[k]\hat{y}_i[k+n_0+j|k]. \tag{22}$$

The design objective is to stabilize the queue length around the set point while keeping the difference between each control move as small as possible. The first goal is obvious, and the reason for the second one would be explained in the later section. With the relation given in (21), we define the objective function as

$$\min\ (Y_{ref} - \bar{Y})^T W_y (Y_{ref} - \bar{Y}) + \Delta U^T W_u \Delta U, \tag{23}$$

where Y_{ref} is the modified reference output based on the reference queue length. The reference queue length is given as q_{ref} and the estimated queue length is $\hat{q}[k+n_0+j]$. It follows that

$$Y_{ref} = \begin{bmatrix} q_{ref} - \sum_{i=1}^{\pi} w_i[k]q_i \\ q_{ref} - \sum_{i=1}^{\pi} w_i[k]q_i \\ \vdots \\ q_{ref} - \sum_{i=1}^{\pi} w_i[k]q_i \end{bmatrix}. \tag{24}$$

Furthermore, $\bar{Y}$ is a vector of the weight estimations of the model outputs over the prediction horizon, while ΔU is a vector of dropping probability increments over the control horizon, i.e.,

$$\bar{Y} = \begin{bmatrix} \bar{y}[k+n_0+1|k] \\ \bar{y}[k+n_0+2|k] \\ \vdots \\ \bar{y}[k+n_0+s|k] \end{bmatrix}, \quad \Delta U = \begin{bmatrix} \Delta p[k] \\ \Delta p[k+1] \\ \vdots \\ \Delta p[k+m] \end{bmatrix}. \tag{25}$$

Finally, W_y and W_u are diagonal weight matrices of queue length differences and control moves, respectively, which can be used to trade-off between the dual objectives of stabilizing the queue and minimizing the control moves.

In (23), the first term minimizes the difference between the real queue length and reference queue length. The second term in the objective function is the penalty for changing the dropping probability. To solve this problem, we need to derive $\bar{y}$ from $\bar{y}[k+n_0+1|k]$ to $\bar{y}[k+n_0+s|k]$. To facilitate this process, the propagation matrix is given as follows.

$$\bar{Y} = M_x^a + M_c^a M_e \Delta U + M_c^a U_0 - M_{cp}. \tag{26}$$

Algorithm 1 The Proposed MAQ Algorithm

Data: Average and variance of service capacity ($\bar{C}$ and σ_C), queue length at each time k
Result: The drop/mark probability $p[k]$ at time k

```
Choose the model number π;
Assign the weight of each model as 1/π;
Define the Kalman Gain matrix L_i and estimation error covariance P_i;
while System is running do
    for i = 1 : π do
        Calculate the estimation ŷ_i^a[k|k];
        Update the Kalman gain L_i and estimation error covariance P_i;
    for i = 1 : π do
        Update probability ρ_i[k] based on (17);
        Calculate weight w_i;
    for j = 1 : n_0 do
        Estimate feature state information x̂_i^a[k + j|k];
    Minimize the objective function (23) and obtain the analytical solution (27);
    Use the first element of the result to obtain p[k];
    Generate uniformly distributed random number u(0, 1);
    if u < p[k] then
        Drop the head-of-line packet at CRN BS queue;
```

The matrices in (26) are given as

$$M_x^a = \sum_{j=1}^{\pi} w_j[k] \begin{bmatrix} Q_j^a(\Phi_j^a) \\ \vdots \\ Q_j^a(\Phi_j^a)^m \end{bmatrix} \hat{x}_j^a[k+n_0|k]$$

$$M_e = \begin{pmatrix} 1 & & 0 \\ \vdots & \ddots & \\ 1 & \cdots & 1 \end{pmatrix}, \quad U_0 = \begin{bmatrix} p[k-1] \\ \vdots \\ p[k-1] \end{bmatrix}$$

$$M_c^a = \sum_{j=1}^{\pi} w_j[k] \begin{bmatrix} Q_j^a\Gamma_j^a & & 0 \\ \vdots & \ddots & \\ Q_j^a(\Phi_j^a)^{m-1}\Gamma_j^a & \cdots & Q_j^a\Gamma_j^a \end{bmatrix}$$

$$M_{cp}^a = \sum_{j=1}^{\pi} w_j[k] \begin{bmatrix} Q_j^a\Gamma_j^a & & 0 \\ \vdots & \ddots & \\ Q_j^a(\Phi_j^a)^{m-1}\Gamma_j^a & \cdots & Q_j^a\Gamma_j^a \end{bmatrix} \begin{bmatrix} p_j \\ \vdots \\ p_j \end{bmatrix},$$

where M_{cp} is the compensation for the deviation from operating point p_i of each model i. The optimization variables of problem (23) is ΔU, which can be solved as

$$\Delta U = (M_e^T M_c^{aT} W_y M_c^a M_e + W_u)^{-1} M_e^T M_c^{aT} \times W_y(\tilde{Y}_{ref} - M_x^a - M_c^a U_0), \tag{27}$$

where $\tilde{Y}_{ref} = Y_{ref} + M_{cp}$.

D. Parameter Tuning and Analysis

The proposed CRN congestion control algorithm is presented in Algorithm 1. In the proposed MAQ scheme, there are several parameters that need to be tuned to achieve good performance. First, the two weight matrices W_y and W_u in the objective function (23) are directly related to the optimal control signal. The control move is the dropping probability in the range of [0, 1]. We can use the saturation function to enforce this range, i.e.,

$$p[k] = \max\{0, \min\{1, p[k-1] + \Delta p[k]\}\}. \tag{28}$$

However, to avoid offset or unstability, it is necessary to choose a large W_u. Usually, the larger the ratio W_y/W_u, the more aggressive the control move. For the CRN congestion control system, we recommend $W_y = \text{diag}(1, 1, \ldots, 1)$ and $W_u = \text{diag}(10^6, 10^6, \ldots, 10^6)$, which work well in our simulation study.

Second, in the estimation procedure, the matrix $P_i[0]$ should be initialized with large and diagonal values if we do not know the covariance of the state variable. Moreover, as the capacity variations may be large and the estimation error may not be zero due to the uncertain nature of the flow model, the disturbance variance O_i is set to be the standard deviation of capacity σ_C and the output variance G_i is set to a small but non-zero value. According to [13], the larger the ratio O_i/G_i, the smaller the response time, but the larger the overshoot (even leading to oscillation). We choose $G_i \in [0.4, 1]$ in this paper based on our simulation study.

Third, for weight calculation, the coefficient λ determines the sensitivity of weights to estimation error. Consider the potentially large network parameter variation, which leads to estimation error, we would not recommend a large λ since it will easily drive the model probability $\rho_i[k]$ to 0. Given such a system with large variations of parameters and conditions, we set $\lambda = 1$ in this paper.

To estimate the time complexity of the proposed algorithm, we first locate the computing bottleneck. Notice that, the time complexity of estimation using Kalman Filter is linear with the prediction horizon m; and the weight calculation is linear to the number of models, which is typically small. Thus, the dominant part of the computation is the final calculation of the control input. The calculation of matrix M_c^a has a complexity of $O(\pi m^2 T_{mp})$, in which T_{mp} is the complexity of two 3×3 matrix multiplication and it's roughly a constant time. The calculation of final ΔU will take $O(T_1 m^3 + T_2 m^2 + T_3 m^2) = O(m^3)$. Since π is normally smaller than m, we can conclude that the time complexity is $O(m^3)$. In general, the time complexity is linear with the number of models and cubic to the prediction horizon, i.e., $O(\pi m^2 + m^3)$.

IV. Simulation Study

In this section, we evaluate the performance of MAQ with NS2 simulations. We extend the NS2 simulator with the multiple-channel extension, TDM module, and the channel bonding/aggregation module. We assume 10 licensed channels: five of them with a capacity of 500 Kb/s and the other five with a 1 Mb/s capacity. As in prior work [6], [10], each channel is modeled as an on-off process. Both the on and off periods are exponentially distributed with mean $\eta(\text{on}) = 4$ seconds and $\eta(\text{off}) = 5$ seconds, respectively. The sensing phase of the

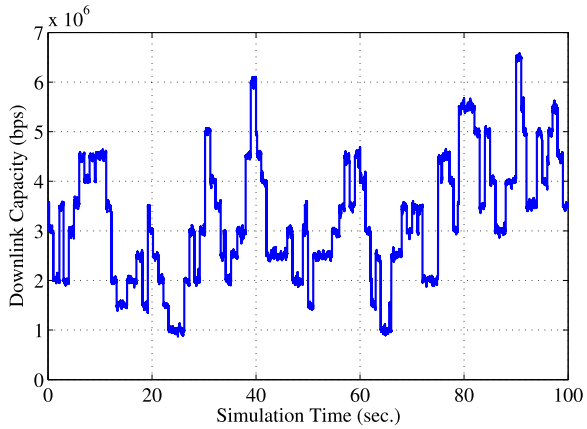


Fig. 3. CRN downlink capacity over time used in one of the simulations.

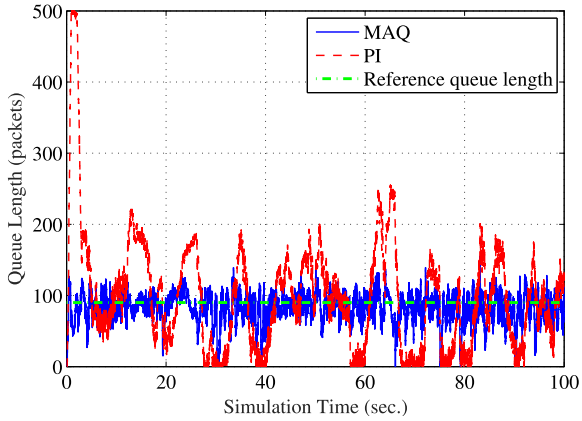


Fig. 4. Queue length dynamics under responsive background traffic.

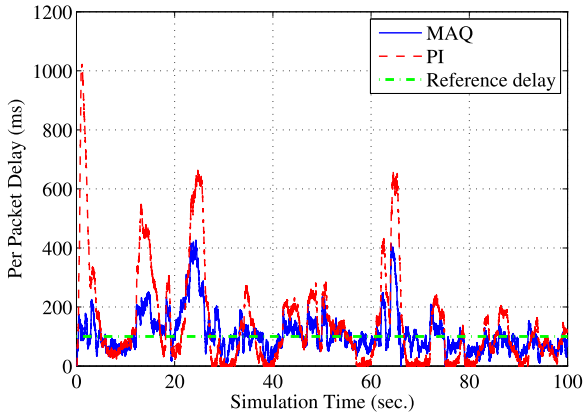


Fig. 5. Packet queuing delay dynamics under responsive background traffic.

CRN is 20ms and the transmission phase is 200ms. If there is no channel available after the sensing period, the frozen state will continue until an idle channel is found.

The CRN maintains a large number of TCP connections. The RTT and propagation delay are a key factor influencing the performance of congestion control. In real networks, different users may have different propagation delays. So we choose propagation delays uniformly distributed in $(100, 150)$ ms (unless stated otherwise). The capacity of the wireline link between the BS to a TCP server is 20 Mb/s. The average capacity for the CRN downlink is $\bar{C} = 3.2$ Mb/s with variance $\sigma_C = 1.3$ Mb/s. The capacity $C(t)$ used in one of the simulations is shown as a function of time in Fig. 3. The packet size is 540 Bytes with 500 Bytes data and 40 Bytes header. The reference queue length is set to 90 packets, corresponding to a queueing delay around 100 ms. The queue size B is set to 500 packets (about 2Mb).

For comparison purpose, we also simulated two existing control-theoretic schemes: (i) the proportional integral (PI) controller, which is canonical and has been shown to be effective on dealing with varying capacities [18]; (ii) the discrete sliding modes (DSM) controller, which is a most recent advance that is based on the principles of discrete sliding-mode control [19]. We set the PI parameters as: 4.839×10^{-5} and 4.346×10^{-5}, as recommended in [18]. The DSM parameter is set as recommended in [19], with $\gamma = 0.01$, and a sigmoid function with $\delta = 10$ is used. We test the proposed scheme under responsive and non-responsive background traffic and under various parameter settings.

A. Responsive Background Traffic

We first conduct simulations with a fixed number of long-lived FTP sessions and HTTP background flows. The HTTP flows are generated using PACKMIME-HTTP in NS2 with a rate of 10, i.e., on average 10 HTTP connections are generated in every second. The transport agent in each source node is set to be TCP/Reno. The number of SUs is $N = 60$. For MAQ, the parameters are set as: $q_i \in \{30, 60, 90, 120, 150\}$ packets, $C_i \in \{1.2, 2.1, 3, 3.9, 4.8\}$Mb/s, $T_p = 100$ms, $W_y = \mathbf{I}$, $W_u = 10^6\mathbf{I}$, where $\mathbf{I}$ is the identity matrix. The covariance parameters are set as $O_i = 300$ and $G_i = 1$, for $i = 1, 2, \ldots, 5$.

Figure 4 presents the queue length dynamics of MAQ and PI. The sudden variation of capacity would cause the queue length change immediately. We find that MAQ can stabilize the queue around the reference point of 90 packets with almost no underflow (i.e., an empty queue), while the PI queue exhibits much larger variations with a lot of underflows. Fig. 5 presents the packet queuing delay dynamics. PI cannot adjust its drop probability quickly enough to adapt to the capacity variations, while MAQ is able to reject the influence of capacity variation and hence achieves lower and more stable queueing delays.

We also present the queue length, delay, and link utilization statistics in Table I. The average queue length of MAQ is closer to the reference point and its standard deviation is about a quarter of that of PI. The average delay of MAQ is also about 20ms lower than that of PI, and its delay standard deviation is 87ms lower than that of PI. It is worth noting that MAQ achieves both lower and more stable queueing delay than PI.

B. Non-Responsive Background Traffic

We next consider a more realistic setting by introducing a non-responsive traffic generator for background traffic.

TABLE I
QUEUE LENGTH, QUEUEING DELAY, AND LINK UTILIZATION STATISTICS

	Average queue length (packets)	STD queue length (packets)	Mean delay (ms)	STD delay (ms)	Average Utilization (%)
Responsive Background Traffic					
MAQ	82	20	87.5	51.9	99.73
PI	98	81	107.4	138.4	96.79
Non-responsive Background Traffic					
MAQ	78	33	72.3	39.8	97.14
PI	140	201	193.5	203.5	41.23

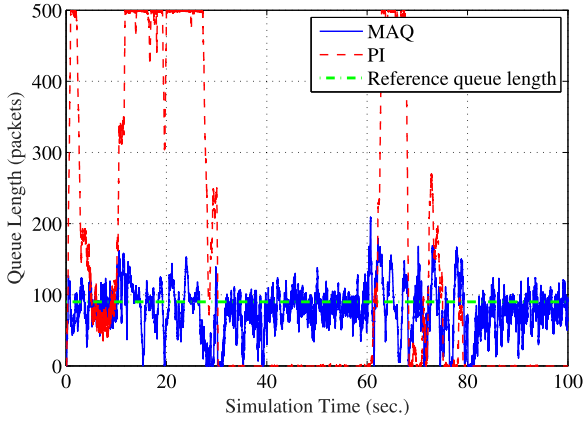


Fig. 6. Queue length dynamics under non-responsive background traffic.

The non-responsive sources would act like disturbance by constantly injecting packets to the queue with exponentially distributed burst time and idle time. We use User Data Protocol (UDP), which does not perform congestion control, for the non-responsive flows. There are 60 FTP flows and 5 non-responsive flows. The burst data rate is 1Mb for each non-responsive flow with packet size 500 Bytes. The non-responsive flows are activated in the following two time intervals: [10s, 30s] and [60s, 80s].

The queue length dynamics with non-responsive background traffic are presented in Fig. 6. Both queues oscillate with a larger range due to the unexpected background traffic bursts. The PI queue overflows (i.e., buffer becomes full) for almost the entire non-responsive traffic transmission period, and then its buffer underflows in the following period when the burst transmissions are off. The MAQ queue, on the other hand, remains relatively stable and takes less time to recover when the bursts are off. We find that since PI has too much overflow and packets drop, the sender's TCP window size would drop to 1, which greatly affects the TCP transmission rate. The dropping probability of PI is also kept high even after the burst transmission period. This is the reason of its long recover time. MAQ, however, can stabilize the queue length around the reference point even with large variations in both the service capacity and background traffic, thus avoiding both buffer underflow and overflow and high link utilization. Another benefit of MAQ is the much lower and more stable packet queuing delay in such highly dynamic situation, as shown in Fig. 7.

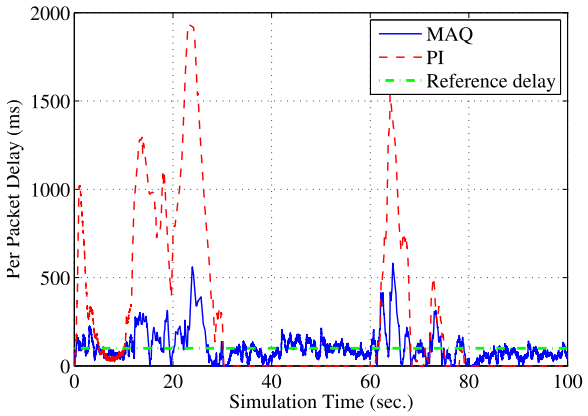


Fig. 7. Packet queuing delay dynamics under non-responsive background traffic.

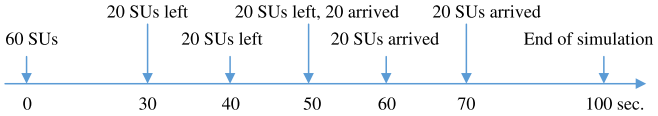


Fig. 8. Change of the number of FTP connections during the simulation period.

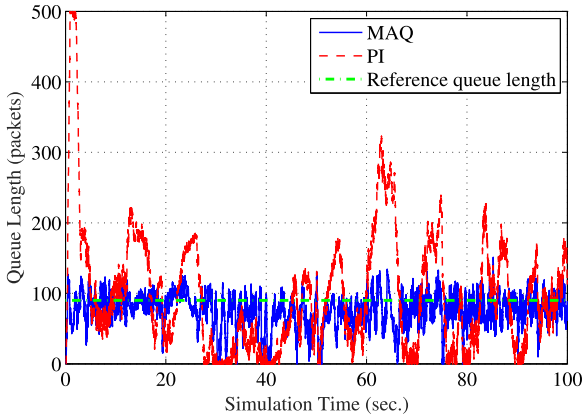


Fig. 9. Queue length dynamics under varying number of connections.

The statistics for the non-responsive background traffic simulations are also presented in Table I. Clearly, MAQ is more robust to strong disturbances in both background traffic and service capacity. In this scenario, the MAQ average delay is only 37.36% of that of PI, and MAQ achieves a 56% gain on link utilization over PI.

C. Varying Number of FTP Connections

We also examine the case when the number of FTP connections varies over time. In this experiment, the simulation starts with 60 FTP connections. The changes of the number of FTP connections during the simulation period are shown in Fig. 8.

The queue length and delay dynamics for this simulation are presented in Figs. 9 and 10. When the connection number goes down, there are fewer TCP flows, which lead to an empty queue. MAQ can act quickly to adjust the dropping

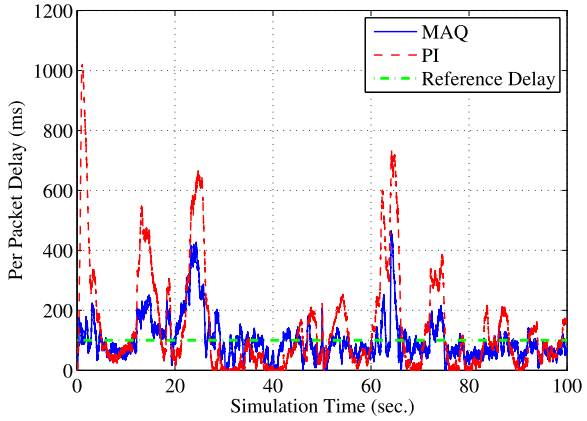


Fig. 10. Packet queuing delay dynamics under varying number of connections.

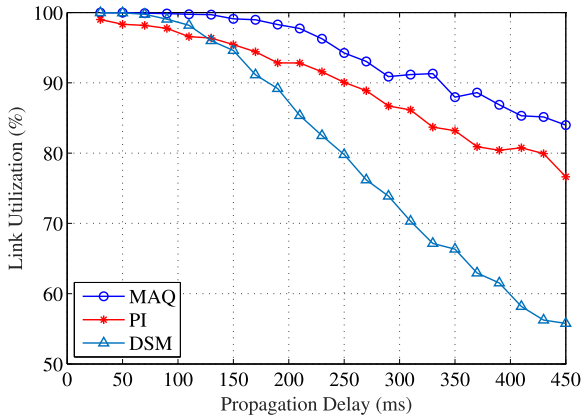


Fig. 11. Average link utilization under increased propagation delays.

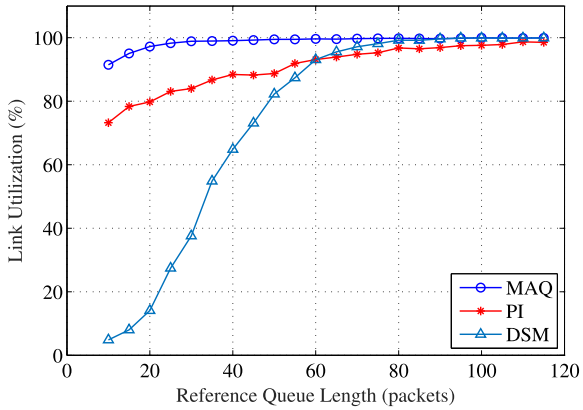


Fig. 12. Link utilization under under increasing reference queue lengths.

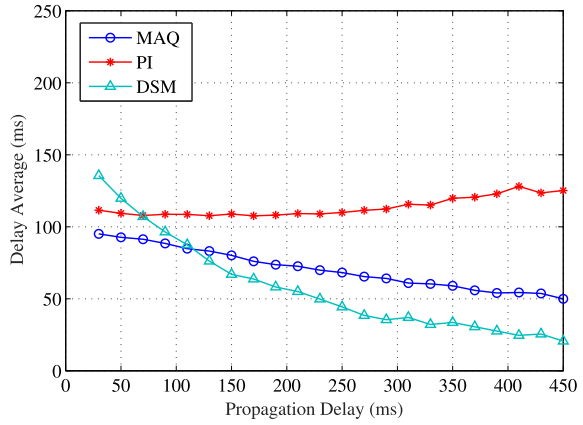


Fig. 13. Average queueing delay under increasing propagation delays.

probability to allow the source congestion window to grow, which will stabilize the queue length again at the reference point. However, the PI queue length will become empty when the connection number goes down, due to its conservative adjustment policy, leading to low link utilization. The superior delay performance of MAQ can also be observed in Fig. 10.

D. Different Propagation Delay and Reference Queue Length

Finally, we analyze the impact of different propagation delay and reference queue length on the congestion control performance. We vary the average propagation delay from 30 ms to 450 ms in steps of 20 ms in a series of simulations. In each simulation, the propagation delay of each SU is the average plus a disturbance uniformly chosen from [0, 50] ms. We also set different reference queue length in the range of [10, 115] packets, which is indicative of different queueing delays. In the simulations, we compare the link utilization and delay of MAQ with that of PI and DSM, which is shown to be robust against system condition variations [19].

Fig. 11 shows the link utilization of each scheme under different propagation delays. A large propagation delay has a negative impact on all the three algorithms, since it is harder to accurately predict future states and outputs. When the RTT is larger than the average capacity changing period, it will be impossible to properly control the queue. This is because when a control signal takes effect on the queue, the service capacity may have already changed to some other value. We find that MAQ can still maintain a considerably higher throughput than both PI and DSM, and the DSM performance degrades more quickly than MAQ and PI under increased propagation delays.

Reference queue length is also critical for congestion control, since a large buffer could mitigate the effect of capacity variation. The link utilization of the three schemes for increased reference queue lengths are presented in Fig. 12. Again, MAQ achieves considerably higher link utilization than the other two schemes, especially when the reference queue length is small. For example, when the reference queue length is 10 packets, the MAQ link utilization is about 20% higher than that of PI and 70% higher than that of DSM. When the reference queue length becomes large, all the three schemes achieve a high link utilization. However, as can be seen later in Fig. 15, PI and DSM achieve the high link utilization at the cost of higher delay and large delay variation than MAQ.

The delay performance statistics, i.e., average and standard deviation of delay, under different propagation delay values and reference queue lengths are plotted in Figs. 13, 14, 15, and 16, respectively. We find that the average delay of MAQ is not only much smaller than that of PI (e.g., around 50% of that of PI), but also much more

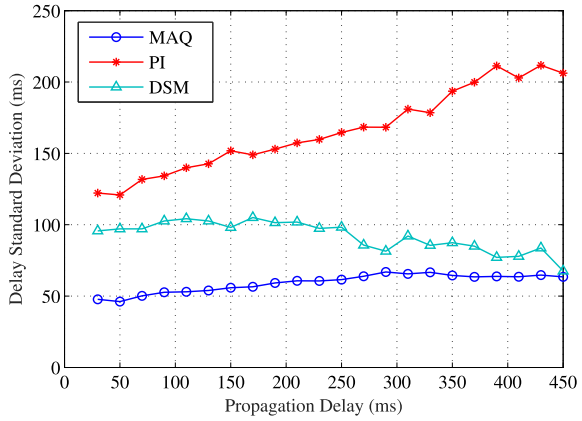


Fig. 14. STD of queueing delay under increasing propagation delays.

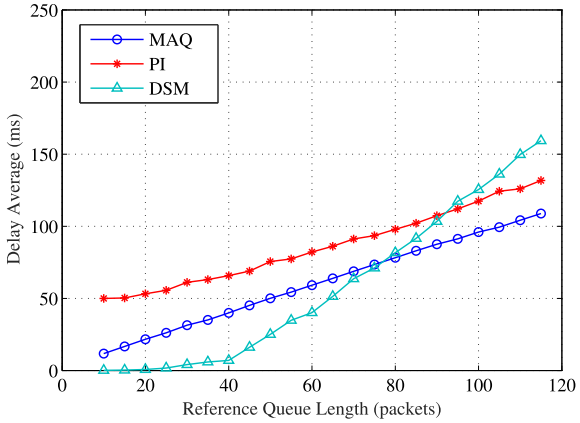


Fig. 15. Average queueing delay under increasing reference queue lengths.

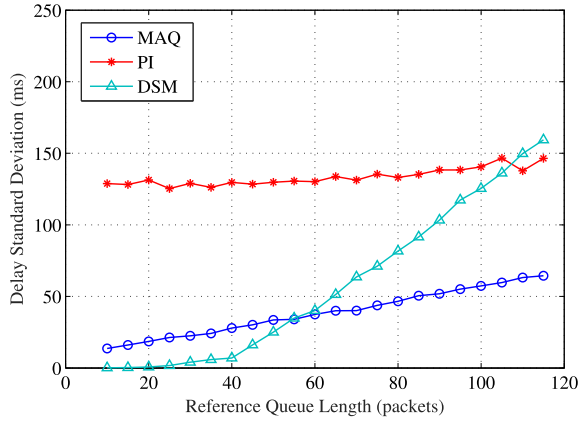


Fig. 16. STD of queueing delay under increasing reference queue lengths.

stable over the entire range of propagation delay values. DSM exhibits better performance on controlling queuing delay than PI. Although its queueing delay is low when propagation delay is large, this is achieved at the cost of a low throughput, since its queue is empty more often than the other two schemes.

Under different reference queue lengths, the average queuing delay goes up roughly linearly since it is around $q_{ref}/\bar{C}$, as shown in Fig. 15. The DSM has very small average delay when the reference queue length is lower than 40 packets.

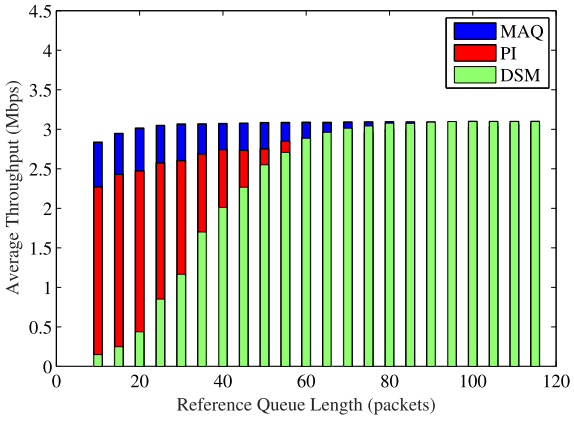


Fig. 17. Average throughput under increasing reference queue lengths.

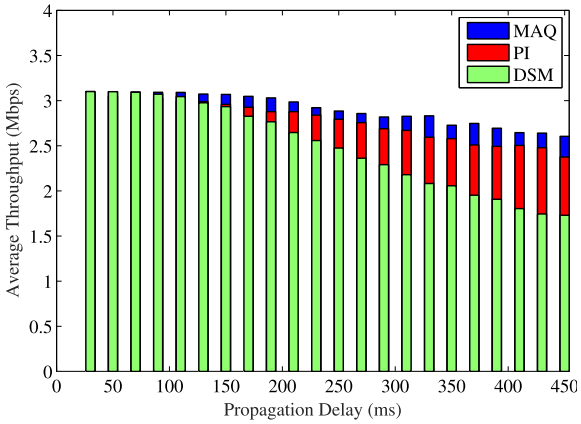


Fig. 18. Average throughput under increasing propagation delays.

As discussed, this is achieved at the cost of very low link utilization (see Fig. 12). MAQ not only achieves low queueing delay and delay variation, the two MAQ curves are also less sensitive to the increased reference queue lengths than the other two schemes.

In order to fully compare these three algorithms with different QoS indexes, we run each simulation 20 times and present the average throughput at the bottleneck link of different schemes with different reference queue lengths and propagation delays in Fig. 17 and Fig. 18, respectively. All the throughput results are bounded by the nominal link capacity, obviously. It's clear that the proposed algorithm outperforms the two benchmark algorithms in all the scenarios. More important, the proposed scheme can maintain an acceptable throughput even under extreme situations such as large propagation delay or small reference queue length.

V. Related Work

In this section, we briefly review related work on congestion control, which can be classified as end-to-end solutions and router-based solutions. We then discuss the several closely related work on congestion control in CRNs.

A. End-to-End Solutions

In the early age, congestion control algorithms are mostly based on the assumption that all packet losses are caused

by buffer overflow at the bottleneck router. TCP Reno [20], TCP Tahoe [21], and TCP NewReno [22] were the several most popular algorithms based on such an idea. On the other hand, protocols were proposed, e.g., TCP Vegas [23], which was delay based. The mainstream protocols that most computer system use are TCP Cubic [24] and Compound TCP [25]. These protocols are mainly based the classical AIMD algorithm and combined with various modifications. However, none of the existing protocols are suitable for congestion control in CRNs, where the service capacity usually has frequent, large variations.

Cai *et al.* [26] proposed a new congestion control scheme, termed semi-TCP, for multi-hop wireless networks based on the request-to-send/clear-to-send handshake. Wang *et al.* [27] proposed a scheme for the configuration of lower-layers, which can achieve an enhancement to the TCP throughput. This work is mainly focused on lower layer configuration. It does not solve the basic problems such as how to deal with the capacity variation due to PU activity and channel fading. One potential problem of cross-layer design is that it requires frequent information exchange, which may waste the invaluable transmission time and introduce large overhead. In addition, none of the existing schemes consider freezing TCP transmissions during the spectrum sensing period.

Most recent congestion control proposals were mainly focused on cellular networks, which also have varying link capacity due to fading and shadowing [28], [29]. The basic idea of [28] was to use a stochastic model: Poisson process with rate that follows a Brownian motion to predict the capacity of the wireless link. This model may not work well in CRN because the variation of capacity is mainly caused by the activity of primary users, in addition to fading and shadowing effects. Yasir *et al.* [29] proposed a method based on the training model between packet delay and window size. However, the most significant difference between cellular network and CRN studied in this paper is, the cellular BS maintains a separate queue for each user (e.g., each user with a dedicate channel), but the CRN BS maintains a shared queue for all users in our model. Thus the basic assumption in such protocols does not hold and the application of such protocols in CRN would be hard, if not impossible. Also the spectrum sensing period will have a big effect on congestion control, which calls for a new CRN specific protocol design.

B. Router-Based Solutions

Our work is closely related to the literature on AQM design in the context of the Internet. Instead of actually dropping packets, the authors in [30] proposed a method with Explicit Congestion Notification (ECN), to provide a feedback from the bottleneck router. The authors in [31] presented a control theoretic analysis of random early detection (RED) [32] and found that RED was sensitive to its parameter setting and it was hard to stabilize if delay was taken into consideration. In [18], a Proportional-Integral (PI) AQM scheme was developed. However, the PI scheme may react slowly to outside disturbances. Model predictive control (MPC) is an important industry control method, which predicts future system states and computes the optimal control input at each sampling time slot. Generalized predictive control has been used in network congestion control [33] and an MPC based AQM method (MPAQM) was proposed in [34]. These methods can compensate the impact of RTT with the model predictor. However, they do not explicitly take disturbance (e.g., varying service capacity) into consideration.

Chen *et al.* [35] proposed a cross-layer congestion control, routing, and scheduling design scheme in Ad Hoc Wireless networks. The authors formulated a resource allocation problem in networks with fixed wireless channels over rate constraint and scheduling constraint, and then decomposed it into three subproblems: congestion control, routing, and scheduling. However, this work did not consider the variation of channel condition and mainly works for Ad Hoc networks. Chavan *et al.* [36] proposed a solution to address the capacity variation problem due to a fading wireless channel with the H_∞ technique. However, its conservative policy makes it unsuitable for CRNs, where the capacity may change at various/fast timescales. In [37], the authors proposed a latency control algorithm, termed CoDel, which provided a solution to the *bufferbloat* problem; but the design did not directly apply to CRNs. In [19], the authors proposed an AQM algorithm, termed DSM, based on the discrete sliding mode control (DSMC) theory. Although it deals with high frequency oscillation of link capacity, DSMC suffers from the over control problem and it is hard to choose a proper control gain for this algorithm.

The AIMD throughput efficiency could also be ensured by tuning the buffer size at the router. One classical rule is setting the buffer size to be equal to the product of bandwidth and average delay (normally RTT) of the flows using this router, called Bandwidth-Delay Product (BDP) [38]. Appenzeller *et al.* [39] claimed that the classical rule is outdated and proposed a new scheme that introduce $\sqrt{n}$ as a scale. Vu-Brugier *et al.* [40] provided a comparison over multiple buffer-sizing strategies and summarized questions of the utility of these policies. Problems such as complex mixed flow connection length and RTT, traffic patterns variation, etc., made it difficult to realize in modern networks, especially in CRNs where the bandwidth varies largely and with mixed flow types. In [41], an adaptive buffer sizing policy for TCP flows in 802.11e WLANs was proposed. The buffer size was updated with the observation of idle and busy times. However, this scheme did not consider the large capacity variation and time delay

C. CRN Congestion Control Schemes

The problem of enhancing TCP performance in CRNs has been addressed in only a few papers. In [42], the authors proposed an cross-layer design scheme for CRN called TCP Everglades (TCPE). The authors proposed a new available bandwidth estimation method based on RTT and RTT_{min} and assumption that the host knows information such as sensing period, etc. However, the design did not consider the time delay and the assumption that host knows everything may not be valid in some cases. In [9], the authors proposed a new

network management framework, termed DSASync, for DSA based WLANs. This work used a zero-size receive window to freeze the TCP sender and smooth the ACKs from the BS during the sensing period. Its design is largely based on heuristic methods, while smoothing ACKs would slow down the growth of window size during the valuable transmission period.

In [11], a cross-layer approach was proposed to maximize throughput by jointly adjusting spectrum sensing, access decision, and modulation and coding scheme in the physical layer. The problem was formulated based on a POMDP framework and solved by dynamic programming. This is an offline scheme that requires full prior knowledge of all system statistics, due to the complex POMDP approach. In [10], the impact of channel sensing and spectrum opportunity change on TCP performance was analyzed and a window based transport control protocol for CR Ad Hoc Networks was proposed. The proposed protocol was mainly based on heuristics rather than a rigorous theoretic analysis, and the enhancements may not be backward compatible with existing TCP protocols that have already been widely used in both wireless and wireline networks.

VI. Conclusion

In this paper, we designed a congestion control scheme for infrastructure-base CRNs. The goal was to stabilize the TCP buffer at the CRN BS under a wide range of system/network parameters and dynamics. The proposed scheme, termed MAQ, was based on MMPC with enhanced state and output estimation. The proposed scheme was evaluated with extensive NS2 simulations under various background traffic types and network/system parameter settings. It was shown to achieve superior performance over two benchmark schemes.

References

[1] M. X. Gong, R. J. Stacey, D. Akhmetov, and S. Mao, "A directional CSMA/CA protocol for mmWave wireless PANs," in *Proc. IEEE WCNC*, Sydney, NSW, Australia, Apr. 2010, pp. 1–6.

[2] Z. He, S. Mao, and T. Rappaport, "On link scheduling under blockage and interference in 60-GHz ad hoc networks," *IEEE Access J.*, vol. 3, pp. 1437–1449, Sep. 2015.

[3] M. Feng and S. Mao, "Harvest the potential of massive MIMO with multi-layer techniques," *IEEE Netw.*, vol. 30, no. 5, pp. 40–45, Sep./Oct. 2016.

[4] Y. Xu, G. Yue, and S. Mao, "User grouping for massive MIMO in FDD systems: New design methods and analysis," *IEEE Access J.*, vol. 2, no. 1, pp. 947–959, Sep. 2014.

[5] Y. Zhao, S. Mao, J. O. Neel, and J. H. Reed, "Performance evaluation of cognitive radios: Metrics, utility functions, and methodology," *Proc. IEEE*, vol. 97, no. 4, pp. 642–659, Apr. 2009.

[6] Q. Zhao and B. M. Sadler, "A survey of dynamic spectrum access," *IEEE Signal Process. Mag.*, vol. 24, no. 3, pp. 79–89, May 2007.

[7] D. Hu and S. Mao, "Streaming scalable videos over multi-hop cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 9, no. 11, pp. 3501–3511, Nov. 2010.

[8] K. Xiao, S. Mao, and J. Tugnait, "Congestion control for infrastructure-based CRNs: A multiple model predictive control approach," in *Proc. IEEE GLOBECOM*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[9] A. Kumar and K. G. Shin, "Managing TCP connections in dynamic spectrum access based wireless LANs," in *Proc. IEEE SECON*, Boston, MA, USA, Jun. 2010, pp. 1–9.

[10] K. R. Chowdhury, M. Di Felice, and I. F. Akyildiz, "TP-CRAHN: A transport protocol for cognitive radio ad-hoc networks," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2482–2490.

[11] C. Luo, F. R. Yu, H. Ji, and V. C. Leung, "Cross-layer design for TCP performance improvement in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 5, pp. 2485–2495, Jun. 2010.

[12] X. Chen, H. Zhai, J. Wang, and Y. Fang, "A survey on improving TCP performance over wireless networks," in *Resource Management in Wireless Networking*, M. Cardeiand, I. Cardei, and D.-Z. Du, Eds. Norwell, MA, USA: Kluwer, 2005, pp. 657–695.

[13] M. Kuure-Kinsey and B. W. Bequette, "Multiple model predictive control strategy for disturbance rejection," *Ind. Eng. Chem. Res.*, vol. 49, no. 17, pp. 7983–7989, Jul. 2010.

[14] H. A. B. Salameh, M. Krunz, and D. Manzi, "Spectrum bonding and aggregation with guard-band awareness in cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 569–581, Mar. 2014.

[15] V. Misra, W.-B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 151–160, Oct. 2000.

[16] T. Lakshman, A. Neidhardt, and T. Ott, "The drop from front strategy in TCP and in TCP over ATM," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Mar. 1996, pp. 1242–1250.

[17] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 689–702, Oct. 2003.

[18] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, Apr. 2001, pp. 1726–1734.

[19] P. Ignaciuk and M. Karbowańczyk, "Active queue management with discrete sliding modes in TCP networks," *Bull. Polish Acad. Sci. Techn.*, vol. 62, no. 4, pp. 701–711, Dec. 2014.

[20] D. Cox and L. Dependence, "A review," in *Statistics: An Appraisal*, H. David and H. David, Eds. Ames, IA, USA: Iowa State Univ. Press, 1984, pp. 55–74.

[21] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, 1988.

[22] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," Tech. Rep. IETF RFC 6582, Apr. 2012.

[23] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 4, pp. 24–35, Oct. 1994.

[24] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[25] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proc. IEEE INFOCOM*, Barcelona, Spain, Apr. 2006, pp. 1–12.

[26] Y. Cai, S. Jiang, Q. Guan, and F. R. Yu, "Decoupling congestion control from TCP (semi-TCP) for multi-hop wireless networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2013, p. 149, 2013.

[27] J. Wang, A. Huang, and W. Wang, "TCP throughput enhancement for cognitive radio networks through lower-layer configurations," in *Proc. IEEE 23rd Int. Symp. Pers., Indoor Mobile Radio Commun.-(PIMRC)*, Sep. 2012, pp. 1424–1429.

[28] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proc. NSDI*, Lombard, IL, USA, Apr. 2013, pp. 459–471.

[29] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 509–522, Oct. 2015.

[30] S. Floyd, "TCP and explicit congestion notification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 8–23, Oct. 1994.

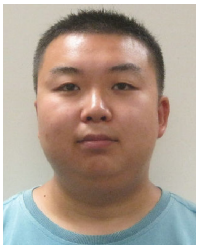
[31] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, Apr. 2001, pp. 1510–1519.

[32] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[33] X.-H. Zhang, K.-S. Zou, Z.-Q. Chen, and Z.-D. Deng, "Stability analysis of AQM algorithm based on generalized predictive control," in *Proc. Int. Conf. Adv. Intell. Comput. Theories Appl.*, 2008, pp. 1242–1249.

[34] P. Wang, H. Chen, X. Yang, and Y. Ma, "Design and analysis of a model predictive controller for active queue management," *Elsevier ISA Trans.*, vol. 51, no. 1, pp. 120–131, Jan. 2012.

[35] L. Chen, S. Low, M. Chiang, and J. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proc. 25TH IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2006, pp. 1–13.
[36] K. Chavan, R. G. Kumar, M. N. Belur, and A. Karandikar, "Robust active queue management for wireless networks," *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 6, pp. 1630–1638, Nov. 2011.
[37] K. Nichols and V. Jacobson, "Controlling queue delay," *ACM Commun.*, vol. 55, no. 7, pp. 42–50, May 2012.
[38] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 45–60, 1994.
[39] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 281–292, 2004.
[40] G. Vu-Brugier, R. Stanojevic, D. J. Leith, and R. Shorten, "A critique of recently proposed buffer-sizing strategies," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 43–48, 2007.
[41] T. Li and D. Leith, "Adaptive buffer sizing for TCP flows in 802.11e WLANs," in *Proc. 3rd Int. Conf. Commun. Netw. China, (ChinaCom)*, Aug. 2008, pp. 7–11.
[42] D. Sarkar and H. Narayan, "Transport layer protocols for cognitive networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM)*. Mar. 2010, pp. 1–6.



Kefan Xiao (S'14) received the B.E. degree in electronic engineering from Xi'an Jiaotong University, Xi'an, China, in 2011, and the M.S. degree in electronic engineering from Shanghai Jiaotong University, Shanghai, China, in 2014. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA. His research interests include TCP congestion control, video streaming, and transmissions.

Shiwen Mao (S'99–M'04–SM'09) received the Ph.D. degree in electrical and computer engineering from Polytechnic University, Brooklyn, NY, USA. He is currently the Samuel Ginn Distinguished Professor with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA. His research interests include wireless networks and multimedia communications. He was a co-recipient of the Best Paper Award from IEEE ICC 2013, IEEE WCNC 2015, IEEE GLOBECOM 2015, and IEEE GLOBECOM 2016. He also received the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communication Systems. He is currently the Chair of the IEEE ComSoc Multimedia Communications Technical Committee. He is currently a Distinguished Lecturer of the IEEE Vehicular Technology Society. He is on the Editorial Board of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE INTERNET OF THINGS JOURNAL, the IEEE MULTIMEDIA, The ACM GetMobile, among others. He was an Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.

Jitendra K. Tugnait (M'79–SM'93–F'94) received the B.Sc. degree (Hons.) in electronics and electrical communication engineering from the Punjab Engineering College, Chandigarh, India, in 1971, the M.S. and E.E. degrees from Syracuse University, Syracuse, NY, USA, in 1973 and 1974, respectively, and the Ph.D. degree from the University of Illinois at Urbana–Champaign in 1978, all in electrical engineering.

From 1978 to 1982, he was an Assistant Professor of Electrical and Computer Engineering with the University of Iowa, Iowa City, IA, USA. He was with the Long Range Research Division, Exxon Production Research Company, Houston, TX, USA, from 1982 to 1989. He joined the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA, as a Professor in 1989, where he currently holds the title of James B. Davis Professor. His current research interests are in statistical signal processing, wireless communications, and multiple target tracking.

Dr. Tugnait was an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE SIGNAL PROCESSING LETTERS, and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, and a Senior Area Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING. He is currently a Senior Editor of the IEEE WIRELESS COMMUNICATIONS LETTERS.