Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Communications 29 (2006) 1963–1978

computer communications

www.elsevier.com/locate/comcom

Adaptive model predictive TCP delay-based congestion control

Mohammad Haeri [a,*], Amir Hamed Mohsenian Rad [b]

[a] *Advanced Control System Lab., Electrical Engineering Department, Sharif University of Technology, Iran*
[b] *Department of Electrical and Computer Engineering, University of British Columbia, Canada*

Received 22 February 2005; received in revised form 7 December 2005; accepted 8 December 2005
Available online 5 January 2006

Abstract

Adaptive Model Predictive Transmission Control Protocol (AMP-TCP) as a new TCP delay-based congestion control algorithm is introduced. Both aspects of design and implementation of the algorithm are described using simulations on the *ns*-2 network simulator. The design stage is composed of two steps. First, a recursive system identification approach is proposed to capture the network delay dynamics from TCP source view. Second, the proposed modeling is employed to develop an adaptive model predictive TCP congestion control strategy in the absence of any explicit congestion notification. The characteristics and performance of AMP-TCP are investigated using several network simulations. Finally a modified version of the AMP-TCP, called Vegas-Like AMP-TCP, is proposed and compared with the well known TCP Vegas algorithm in term of providing the desired number of buffered in-flight packets in the network.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Congestion control; Transmission Control Protocol; Recursive system identification; Model predictive control; Adaptive control

1. Introduction

Congestion control is a distributed algorithm to share network resources among competing sources. It consists of a source algorithm (e.g., Reno, Vegas, etc.), that dynamically adjusts source rates based on congestion in their paths, and a link algorithm (e.g., DropTail, RED, REM, etc.), that updates, implicitly or explicitly, a certain congestion measure at each link and feeds it back, implicitly or explicitly, to sources that use the link. Different protocols use different metrics as congestion measure. Generally speaking, there are two important congestion measures: *packet-loss* and *packet-delay* [1]. Loss-based (e.g., Tahoe, Reno) and delay-based (e.g., Vegas, FAST) TCP congestion control algorithms use packet-loss and packet-delay measurements respectively to adjust the TCP congestion window size. As pointed out in [2], in the absence of an explicit feedback, delay-based algorithms become the preferred approach for end-to-end congestion control as networks scale up in capacity. In addition, the experimental studies presented in [3,4] have shown that TCP Vegas as a delay-based congestion control strategy achieves a better performance in comparison with TCP Reno as a loss-based congestion control strategy.

In the present work, a novel TCP delay-based congestion control algorithm is proposed using system and control theory. First, the network is defined as a discrete system with dynamical behavior and measurable input and output signals. Then a recursive prediction-error system identification algorithm is employed to model the network delay dynamics. As the next step, the model-based predictive control theory [5] is employed to develop an adaptive model predictive congestion controller. Finally, the original AMP-TCP is modified to adjust the source sending rate in order to keep a certain number of packets buffered in the routers along the routing path. It is actually the same as what TCP Vegas tries to do; however it uses a completely different algorithm. We called the modified version as Vegas-Like AMP-TCP. By implementing AMP-TCP in *ns*-2 network simulator [6], several simulations are performed to investigate the control performance. Fur-

* Corresponding author. Tel.: +98 2166165964; fax: +98 2166023261.
E-mail addresses: haeri@sina.sharif.edu (M. Haeri), hamed@ece.ubc.ca (A.H. Mohsenian Rad).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2005.12.004

thermore, the Vegas-Like AMP-TCP is compared with TCP Vegas and it is observed that the performance improves significantly.

The paper is organized as follows. Section 2 overviews the TCP–Network interaction and defines a new closed-loop delay-based congestion control problem using systems and control theory. In Section 3, the recursive system identification technique is used to model the network's delay dynamic. The AMP-TCP congestion controller, and its modified version, the Vegas-Like AMP-TCP are proposed in Sections 4 and 5, respectively. Extensive ns-2 simulation studies are discussed in Section 6. Finally the paper is concluded in Section 7.

2. Problem formulation

2.1. System theoretical approach

Consider a conventional TCP packet transmission process over the Internet as in Fig. 1. There are three basic components: (1) The sender end-host (the source) which sends data packets one-by-one by known transmission rates and receives the acknowledgement of each acknowledged packet after a measurable delay, called round-trip time (RTT). (2) The receiver end-host (the destination) which receives the mentioned data packets and sends back an acknowledgement for each one. (3) The Internet network which is the carrier of the packet transmission. It has an unknown internal and dynamic behavior [7].

In the source computer, TCP is dynamically dealing with the packet transmission. Let's define it as a system, called *TCP system*. We can then define the rest of the network as another system, called *Network system*.

A TCP source keeps a variable called *congestion window size*, which determines the maximum number of out sending packets that have been transmitted but not yet acknowledged. In packet transmission process, when the congestion window is drained up, the source must wait for an acknowledgment before sending a new data packet. In this way, size of the window controls the source rate (roughly one window of packets is sent every round-trip time [1]). In terms of systems and control theory, the TCP system *excites* the network system by adjusting its window size and consequently its packet transmission rate.

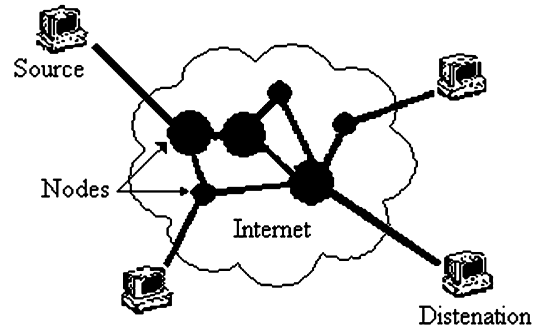

Fig. 1. Basic components of a conventional Internet transmission process.

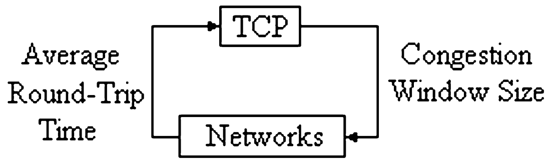

Fig. 2. TCP–Network interaction with system theoretical approach.

Thus, we can define the size of the congestion window as the Network's input variable. It is apparently measurable by the TCP source.

As long as the transmitted packet and its acknowledgement are not lost in the network, the TCP source receives the acknowledgement after a measurable round trip time. The measured round trip time actually indicates the network congestion status which is fed back to the TCP source. It can be interpreted as the network *response*. Note that the congestion window size indicates the *average* packet transmission rate, not its instantaneous rate [1]; thus the effect of any change in the congestion window size can only be observed properly by considering *average* of the measured round-trip times of one window of packets. As a result, we reciprocally use the *average round-trip time* ($\overline{rtt}$) as the network's output variable. The TCP–Network interaction from the system theoretical point of view is depicted in Fig. 2.

2.2. Delay-based closed-loop congestion control problem

Most of the prior studies on congestion control from the systems and control theoretical point of view have dealt with the designing of appropriate *link algorithms*. These algorithms are implemented as the congestion controllers in the routers [8–11] while the rest of the system from the *router's view* is considered as the *plant*. However, our study is dealing with designing the *source algorithm* to be implemented as the congestion controller in the TCP sources. As a result, we need to define a different control problem.

Back to the TCP–Network interaction in Fig. 2, we need to define the plant, the controller, the control and feedback signals and the control goal. They are shown in Fig. 3. In this illustration, TCP is the congestion *controller* and adjusts the congestion window size as the *control signal*. The network is the *plant*. Its congestion measure is the averaged packet round-trip time. This is actually the *feedback signal* in the closed-loop system. The *control goal* is to design a TCP controller to generate the control signal (*cwnd*) such that the plant's output ($\overline{rtt}$) tracks the desired *set point* ($\overline{rtt}_d$).

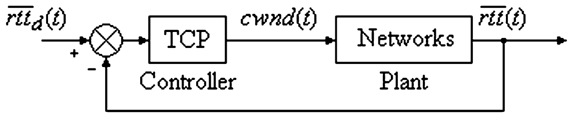

Fig. 3. Proposed delay-based closed-loop congestion control problem.

In the rest of the paper we will discuss on

- How to derive the plant model.
- How to design the controller system.
- How to select the set point to control the network's congestion in a distributed manner.

3. Modeling the network's delay dynamic

The network system (plant) has a dynamic behavior [7], which is unknown from TCP source's view. Its dynamic is actually *time-variant* and depends on factors like the assigned routes, link capacities and background traffic [12,13]. We are to use the periodically measured network's input and output to recursively model its delay dynamic. The modeling tool is the recursive system identification algorithm based on the model prediction error [14]. The *once per RTT sampling* [3] is also used in which the measured data is picked up once within each RTT interval.

Suppose that t_1 and t_2 are two successive sampling times in which the acknowledgements of the k_1^{th} and k_2^{th} packets are received. The averaged round-trip time at t_2 is computed as follows:

$$\overline{rtt}(t_2) = \sum_{k=k_1}^{k_2} rtt(k)/(1 + k_2 - k_1), \tag{1}$$

where $rtt(k)$ is the measured round-trip time of the k^{th} packet and $\overline{rtt}(t_2)$ is the sampled averaged round-trip time at t_2. The sampled congestion window size at t_2, which is showed by $cwnd(t_2)$, is the size of the congestion window at t_2. It does not change during the time between t_1 and t_2. According to the above explanation, the input and output signals of the plant model are defined as follows:

$$\begin{aligned} u(t) &\overset{\Delta}{=} \text{congestion window size at time } t = cwnd(t), \\ y(t) &\overset{\Delta}{=} \text{average packet round trip time at time } t = \overline{rtt}(t). \end{aligned} \tag{2}$$

Since the plant exhibits nonlinear and time varying behavior, to compensate for model/plant mismatches as well as to eliminate the slowly time varying disturbances, the varying values of the variables, i.e., $\Delta u(t)$ and $\Delta y(t)$, are considered in the identification process.

$$\begin{aligned} \Delta u(t) &= u(t) - u(t-1) = cwnd(t) - cwnd(t-1), \\ \Delta y(t) &= y(t) - y(t-1) = \overline{rtt}(t) - \overline{rtt}(t-1). \end{aligned} \tag{3}$$

The above data pairs are used to estimate the parameters of an ARX model structure [14] given in (4).

$$\begin{aligned} &A(q)\Delta y(t) = B(q)\Delta u(t) + e(t) \\ &A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}, \\ &B(q) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}. \end{aligned} \tag{4}$$

where $e(k)$ is the measurement noise, and q^{-1} is the delay operator; i.e., $q^{-1}u(t) = u(t-1)$. The numbers n_a and n_b are orders of the respective polynomials. In most parts of our study the model orders are $n_a = n_b = 1$, so that the ARX model is in the form of (5).

$$\Delta y(t+1) + a\Delta y(t) = b\Delta u(t) + e(t+1). \tag{5}$$

In prediction error system identification algorithms, the model parameters are estimated to minimize an appropriate *weighted prediction error loss function*. It is generally defined in the form of the weighted error squares:

$$\begin{aligned} V(\theta, t) &= \sum_{k=1}^{t} \beta(k)\varepsilon^2(k) \\ &= \sum_{k=1}^{t} \beta(k)(y(k) - \varphi^T(k)\theta)^2, \quad 0 \leqslant \beta(k) \leqslant 1, \end{aligned} \tag{6}$$

where $\varphi(k)$ and θ are the regression and parameter vectors at k^{th} sample time.

$$\varphi(k) = \begin{bmatrix} -\Delta y(k) \\ \Delta u(k) \end{bmatrix}, \quad \theta = \begin{bmatrix} a \\ b \end{bmatrix} \tag{7}$$

$\varepsilon(k)$ is the prediction error (the difference between the model's and plant's outputs) at k^{th} sample time. $\beta(k)$ is the weighting factor and determines how much the k^{th} sample prediction error is important. We use a typical weighting method, called *exponential forgetting* or *exponential discount*:

$$\beta(k) = \lambda^{t-k}. \tag{8}$$

Parameter $\lambda (0 < \lambda < 1)$ is the *forgetting factor*: the newest data get unit weight and the data which is $n = t - k$ sample times old, gets a lower weight of λ^n. Therefore, the old data is forgotten exponentially. The larger λ is selected for the case that the system is time invariant or slowly varying. For time varying systems with fast variation the smaller λ would be appropriate. In control applications, λ is generally selected between 0.95 and 0.999 [15].

The model parameters (θ) are updated at each sample time t to minimize the weighted prediction error loss function:

$$\theta = \arg\min V(\theta, t) = \sum_{k=1}^{t} \lambda^{t-k}(y(k) - \varphi^T(k)\theta)^2. \tag{9}$$

The updating algorithm is discussed in detail in [14] and is as in (10).

$$\begin{aligned} \theta(t) &= \theta(t-1) + L(t)[\Delta y(t) - \varphi^T(t)\theta(t-1)], \\ L(t) &= \frac{P(t-1)\varphi(t)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}, \\ P(t) &= \frac{1}{\lambda}\left[P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)}\right]. \end{aligned} \tag{10}$$

The following initial values are applied in general.

$$\theta(0) = \theta_0, \quad P(0) = P_0 = \alpha I, \tag{11}$$

where θ_0 is an arbitrary vector, I is the unity matrix, and α is a sufficiently large number.

4. Adaptive model predictive TCP delay-based congestion control

In this section, the recursively identified ARX model in (5) is used to develop an appropriate adaptive model predictive congestion controller in TCP.

Assume the following three vector definitions.

$$\begin{aligned} U &= [u(t), u(t+1), \ldots, u(t+M)] \\ &= \text{Future control signals}, \\ Y &= [y(t), y(t+1), \ldots, y(t+P)] \\ &= \text{Future plant output signals}, \\ Y_d &= [y_d(t), y_d(t+1), \ldots, y_d(t+P)] \\ &= \text{Future desired plant output signals}. \end{aligned} \tag{12}$$

In model-based predictive control or simply model predictive control (MPC) theory [5], the goal is to compute the appropriate future control signals (U) such that the plant future output signals (Y) track the desired future output signals (Y_d). In (12), M and P are called *control* and *prediction horizons*, respectively. Since the actual future outputs of the plant are not available, the *model* is used to *predict* the future outputs and the predictions are employed instead of the actual values. The control signal is computed by optimizing an appropriate *cost function*.

In the present study, the control as well as the prediction horizons are set to one ($M = P = 1$) and the model is the recursively obtained ARX model of (5). In the most cases the ARX model order is $n_a = n_b = 1$ and therefore, the plant output is predicted as follows:

$$\Delta\hat{y}(t+1) = -a\Delta y(t) + b\Delta u(t) + d(t+1) \tag{13}$$

$d(t+1)$ represents the unknown (unpredictable) part of the plant output. The desired output of the plant is defined as $\Delta y_d(t+1)$ and is related to the set point of $\overline{rtt}_d$ by (14).

$$\Delta y_d(t+1) = \overline{rtt}_d(t+1) - \overline{rtt}(t). \tag{14}$$

The cost function is selected as the well-known *quadratic function* given by the following equation.

$$J = (\Delta Y_d - \Delta\hat{Y})^T(\Delta Y_d - \Delta\hat{Y}) + \Delta U^T R \Delta U, \tag{15}$$

where R is a positive definite weighting matrix. Because of using unitary control and prediction horizons, this matrix is a positive scalar and is substituted by r as follows:

$$J = (\Delta y_d - \Delta\hat{y})^2 + r\Delta u^2. \tag{16}$$

Remark 1. The cost function in (16) has two terms. The first term is used to minimize the set point tracking error and the second term is used to penalize the $\Delta u(t)$ variations which in turn restrict the *deviations* of the congestion window size. There is a trade off between achieving these two goals. It is determined by adjusting the weighting parameter r.

Remark 2. In a common congestion control problem there is no need to apply the "cheap control strategy". The congestion window size can be increased freely up to its defined maximum; however, the deviations of the congestion window size should be restricted for the reason described in the next remark.

Remark 3. Low deviations in the congestion window size improve the performance of the recursive system identification algorithm. Note that the network is a nonlinear dynamic system and the recursive system identification is applied to fit a linear model around the specific operating point. High deviation in the congestion window size immediately changes the operating point and degrades the modeling performance. On the other hand, low deviation gradually changes the operating point so that the recursive system identification algorithm has enough time to cope with the new network dynamics.

The cost function in (16) is a convex function of $\Delta u(t)$, thus there is a global optimum. To reach the optimal $\Delta u(t)$, it suffices to make the derivative of J with respect to $\Delta u(t)$ equal to zero [16].

$$\frac{\partial J}{\partial \Delta u(t)} = 2(-a\Delta y(t) + b\Delta u(t) + d(t+1) - \Delta y_d(t+1))(b) + 2r\Delta u(t) = 0. \tag{17}$$

This makes the following relation among the known and unknown signals.

$$\Delta u(t)[r + b^2] = b[\Delta y_d(t+1) + a\Delta y(t) - d(t+1)]. \tag{18}$$

And finally the control signal variation is determined as

$$\Delta u(t) = b\frac{\Delta y_d(t+1) + a\Delta y(t) - d(t+1)}{r + b^2}. \tag{19}$$

The congestion window size is updated once per RTT according to the following equation.

$$cwnd(t+1) = cwnd(t) + \Delta u(t). \tag{20}$$

There are two upper and lower bound restrictions on the congestion widow size that should be satisfied. The lower bound is usually two. The upper bound is generally assigned during the connection establishment process. These constraints are formulated as follows:

$$\overline{cwnd} \leqslant cwnd(t) \leqslant \underline{cwnd}. \tag{21}$$

Therefore, the congestion window size determined by the controller is modified according to the following rules.

$$\begin{aligned} &\text{if } cwnd(t) > \overline{cwnd}, \text{ then } cwnd(t) = \overline{cwnd}, \\ &\text{if } cwnd(t) < \underline{cwnd}, \text{ then } cwnd(t) = \underline{cwnd}. \end{aligned} \tag{22}$$

Eqs. (19)–(22) update the congestion window size based on the model predictive control theory. As the model is determined from time variant system identification algorithm in (10), the designed control strategy is categorized in the *indirect adaptive control* schemes [15]; therefore the proposed TCP controller is called *Adaptive Model Predic-*

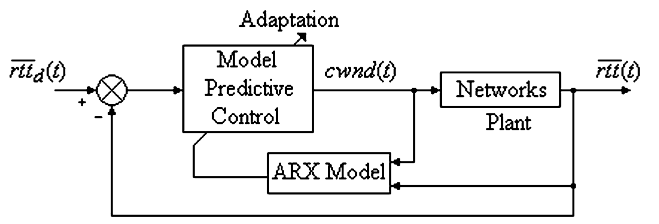

Fig. 4. Proposed AMP-TCP delay-based congestion control.

tive Transmission Control Protocol or AMP-TCP. The closed-loop control system based on AMP-TCP is illustrated in Fig. 4.

Remark 4. Using direct method of Lyapunov stability theorems, one can analyze the closed loop stability of the system. Assuming that the identified model is accurate (nominal case $d(t+1) = e(t+1)$), the discrete *closed-loop transfer function* of the small signal approximation can be derived from (5) and (19).

$$\begin{aligned}\Delta y(t+1) = & -a\ \Delta y(t) \\ & + b\left(b\frac{\Delta y_d(t+1) + a\Delta y(t) - e(t+1)}{r+b^2}\right) \\ & + e(t+1) \\ = & -a\left(1-\frac{b^2}{r+b^2}\right)\Delta y(t) + \left(\frac{b^2}{r+b^2}\right)\Delta y_d(t+1) \\ & + \left(1-\frac{b^2}{r+b^2}\right)e(t+1) \\ = & -a\frac{r}{r+b^2}\Delta y(t) + \frac{b^2}{r+b^2}\Delta y_d(t+1) \\ & + \frac{r}{r+b^2}e(t+1). \end{aligned} \tag{23}$$

Using the following parameters definitions

$$k_1 = -a\frac{r}{r+b^2}, \quad k_2 = \frac{b^2}{r+b^2}, \quad k_3 = \frac{r}{r+b^2} \tag{24}$$

the closed-loop transfer function would be

$$\Delta y(t) = \frac{k_2}{1-k_1q^{-1}}\Delta y_d(t) + \frac{k_3}{1-k_1q^{-1}}e(t). \tag{25}$$

Since r is a positive scalar, we have

$$\frac{r}{r+b^2} < 1. \tag{26}$$

If the identified model in (5) is stable during the TCP connection time, its pole is inside the unit circle; therefore we have

$$|a| < 1. \tag{27}$$

Combining (25) and (26) the following inequality is satisfied.

$$|k_1| = \left|a\frac{r}{r+b^2}\right| < 1. \tag{28}$$

Thus, the small-signal approximation of the closed-loop system in (25) is stable and the following proposition is in order.

Proposition 1. *If the recursive system identification algorithm converges to an accurate and stable model, the small-signal approximation of the closed-loop system is stable.*

5. Vegas-Like AMP-TCP

The closed-loop congestion control problem in Fig. 2 was solved in Sections 3 and 4 by introducing the AMP-TCP protocol. AMP-TCP was actually designed to properly adjust the congestion window size in an effort to make the average RTT track a desired set point; however our third research question has not been answered yet: *How to select the AMP-TCP set point?*

The answer should satisfy two network conditions, especially in larger network topologies. First, the set points of the different AMP-TCP sources should be *compatible*, without any incoherency and second, the set points in different AMP-TCP sources should share the network resources *fairly*.

The authors believe that there are several possible approaches to select the desired set point in each AMP-TCP source. As a proper candidate we borrow the idea behind the well-known TCP Vegas congestion control algorithm and slightly modify the original AMP-TCP.

5.1. TCP Vegas and the estimation of queued packets

TCP Vegas [3] was introduced in 1994 as an alternative to TCP Reno. It improves upon each of the three mechanisms of TCP Reno. The first enhancement is a more prudent way to grow the window size during the initial use of slow-start and leads to fewer losses. The second enhancement is an improved retransmission mechanism where time-out is checked on receiving the first duplicate acknowledgment, rather than waiting for the third duplicate acknowledgment (as Reno would), and leads to a more timely detection of the loss. The third enhancement is a new congestion avoidance mechanism that corrects the oscillatory behavior of Reno and has the specific interest in our study. Vegas' strategy is to adjust the source's sending rate (window size) in an attempt to keep a small number of packets buffered in the routers along the path typically between $\alpha = 1$ and $\beta = 3$ so as to take advantage of extra capacity when it becomes available [17]. To do so, the Vegas source *estimates* the number of buffered (queued) packets as follows:

$$q = cwnd \times \frac{d_q}{\overline{rtt}} = cwnd \times \frac{\overline{rtt} - d_p}{\overline{rtt}} = cwnd\left(1 - \frac{d_p}{\overline{rtt}}\right), \tag{29}$$

where q is the estimated number of queued packets, *cwnd* is the window size, d_q is the queuing round-trip delay and d_p is the propagation round-trip delay which is estimated by getting the minimum measured round-trip time. In *ns-2* implementation, q is rounded to an *integer* number as follows:

$$q = \text{int}\left[cwnd\left(1 - \frac{d_p}{\overline{rtt}}\right) + 0.5\right]. \tag{30}$$

There have been several research studies on the performance analysis and improvement of TCP Vegas. Generally speaking, we can divide them to two categories: (1) analysis of the *steady-state behavior* and (2) analysis of the *transient behavior*. When the steady-state behavior is the concern, the problems like fairness among competitive connections, or performance degradation because of backward congestion or multiple congested links are mainly investigated [18–21]. For example in [18], an aided congestion avoidance mechanism for TCP Vegas, called Aid-Vegas, was proposed. It uses the relative one-way delay of each packet along the forward path to distinguish whether congestion occurs or not. It can effectively solve the backward congestion and unfairness problem of TCP Vegas. On the other hand, when the transient behavior is the concern, the problems like convergence time and initial window size fluctuations are considered [22,23]. In [22], the TCP Vegas is modified to change from slow-start phase to congestion avoidance phase too early; therefore it would not send bursts of packets during its slow-start phase due to the fast increase of window size and will prevent some fluctuations. The authors in [23] also modified the slow-start phase of TCP Vegas to achieve faster convergence. There is no study on the transient behavior of the TCP Vegas congestion avoidance phase yet.

All the prior research works, in both categories, have proposed heuristic "modifications" on TCP Vegas. In contrast, we are going to completely "replace" TCP Vegas by our proposed Vegas-like AMP-TCP algorithm. The proposed Vegas-Like TCP has actually the same goal of TCP-Vegas congestion avoidance phase and tries to adjust the source's sending rate to keep a certain number of packets buffered in the routers along the path. However, the AMP-TCP does the job by an absolutely different algorithm. More precisely, our proposed strategy takes the advantages of mathematical modeling and analytical control theory. To the best of our knowledge, it is the first paper that replaces the TCP Vegas by a TCP congestion control algorithm which completely comes from the systems and control theory. Apart from that, many of the proposed corrections on TCP-Vegas, e.g., modification of the slow-start phase, are still applicable to the Vegas-Like AMP-TCP.

5.2. Vegas-Like AMP-TCP to act as TCP Vegas

To keep the desired (small) number of buffered packets along the routing path, the estimated number of queued packets can be used to determine the desired set point. To do so, (30) can be used to set $\overline{rtt}_d$ as a function of desired number of queued packets which is shown by q_d.

$$\overline{rtt}_d = \frac{d_p}{1 - q_d/cwnd}. \tag{31}$$

If the desired number of queued packets (q_d) is equal to or more than the congestion window size ($cwnd$), $\overline{rtt}_d$ will be infinite or negative; therefore, the congestion window size adjustment algorithm is augmented by the following rule.

$$if\ cwnd \leqslant q_d \text{ then } cwnd = q_d. \tag{32}$$

Applying the mentioned changes, the closed-loop system in Fig. 4 is modified as in Fig. 5. This implies that instead of selecting $\overline{rtt}_d$, q_d should be selected which is done as in the TCP Vegas approach. $\overline{rtt}_d$ is then determined from (31).

In the case of multiple AMP-TCP sources, the basic AMP-TCP and the Vegas-Like AMP-TCP can be compared as illustrated in Figs. 6A and B.

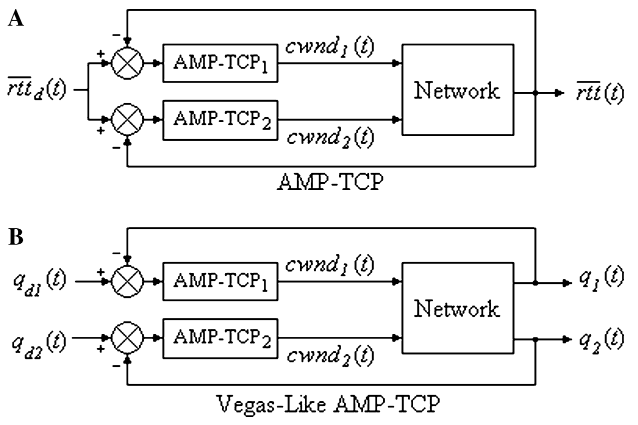

Fig. 6. Multiple-source closed-loop delay-based congestion control.

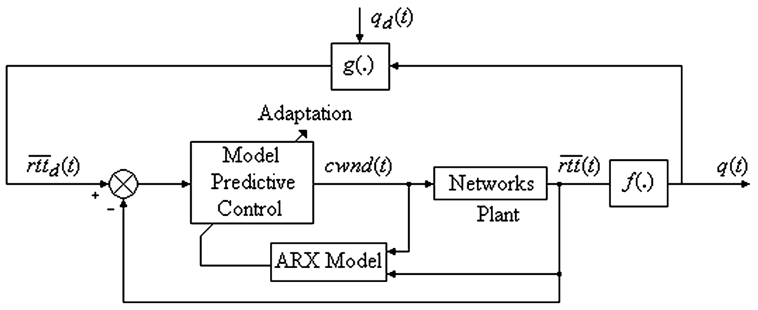

Fig. 5. Vegas-Like AMP-TCP delay-based congestion control.

Remark 5. In the multiple Vegas-Like AMP-TCP congestion control system with N_s AMP-TCP sources, there are N_s single-variable closed-loop systems. In the i^{th} closed-loop system, there is only one control signal which is $cwnd_i$. The other congestion window sizes are viewed as unknown disturbance and noise signals and their effects is formulated as $e(t)$ in (5).

Remark 6. As pointed out in [2], a desired delay-based TCP congestion control algorithm is the one that exploits delay as the congestion measure and is augmentable with the loss information. The proposed AMP-TCP possesses the above property. This algorithm exploits the averaged round-trip time as the congestion measure. The loss-based augmentation can be implemented into the algorithm although it has not been done in the present work.

6. Simulation results and discussions

To investigate the performance, AMP-TCP and Vegas-Like AMP-TCP are implemented in *ns-2* network simulator. A simple but fundamental single-source single-node network topology is simulated in Sections 6.1–6.4 which contains one bottle-neck node, one AMP-TCP source and one *random* Constant Bit Rate (CBR) source. The multiple-source single-node scenario with Vegas-Like AMP-TCP is simulated in Sections 6.5 and 6.6. In each part, a couple of remarks are discussed to analyze the observations. The remarks are mainly based on the provided simulation results; however they are also confirmed by the extra simulations that are not provided because of the space limitations. The complete set of the simulation studies and the AMP-TCP source codes are available in [24].

In the single-source single-node scenario, the random CBR source is persistently submitting UDP packets with average rate of 100 kbyte/s to make the network *background traffic*. All data packets (TCP and UDP packets) have same fixed size of 1000 bytes and the propagation delay in all links is 10 ms. Since the delay-based control performance is going to be investigated, the bottleneck queue size is set to a sufficiently large number. This implies that there would be no packet-loss. The only available congestion measure is the packet-delay; more precisely the averaged round-trip time.

6.1. Open-loop system identification

In this simulation, the congestion window size has been adjusted in the form of a *pseudo random* signal to excite the network system (Fig. 7). The system's input and output signals are gathered by once per RTT sampling strategy and during 100 s simulation. The ARX model parameters are estimated using the forgetting factor of 0.95, 0.98, 0.995, and 0.999. The estimated parameters when the forgetting factor is 0.95 and 0.995 are given in Fig. 8.

Remark 7. By increasing the forgetting factor, the deviation of the estimated parameters decreases; however they converge slowly to their steady state. As the system is time variant, the forgetting factor should be selected small. In the future simulations, the forgetting factor will be fixed on 0.995 that provides the acceptable trade off between estimation accuracy (small deviations) and estimation rapidity (fast convergence).

The measured versus predicted outputs $(\Delta y, \Delta\hat{y})$ as well as the prediction error are illustrated in Fig. 9 for the last 50 samples when $\lambda = 0.995$. It is observed that the simple model-based predictor of (13) provides a satisfactory performance. The open-loop modeling performance can be further improved by using higher model orders. However, it is obvious that the higher model orders require longer updating time. To compare different model orders in the simulated open-loop system identification approach, a convenient metric is the Akaike Final Prediction Error (*FPE*) criterion which is defined as follows [14].

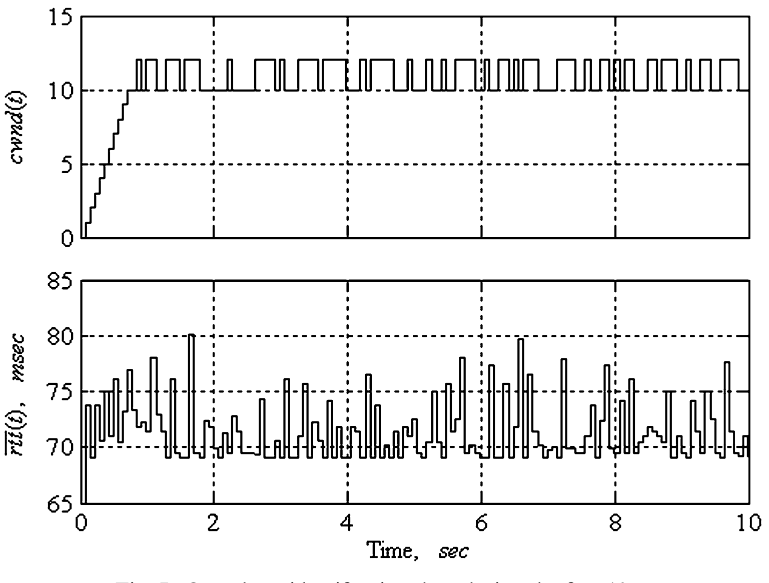

Fig. 7. Open-loop identification data during the first 10 s.

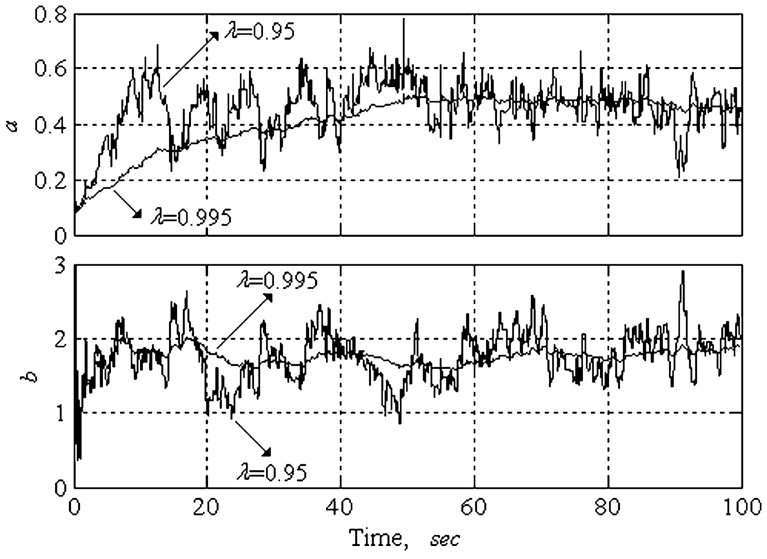

Fig. 8. Estimated model parameters using two different forgetting factors.

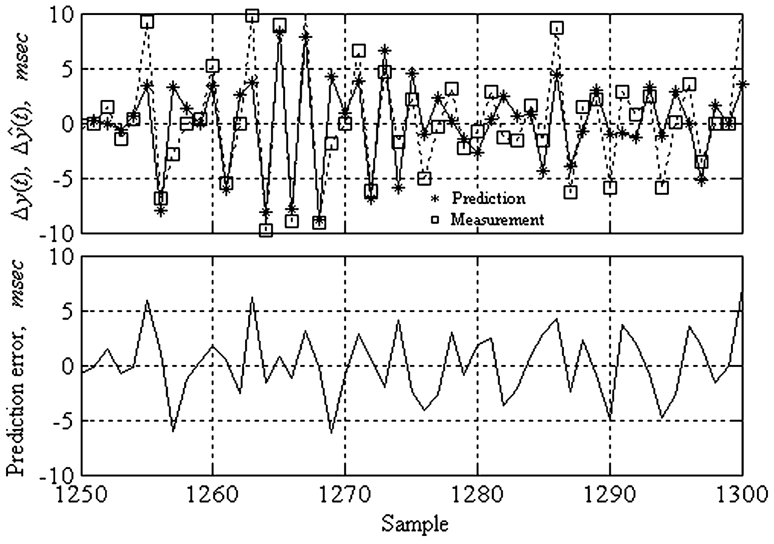

Fig. 9. Open-loop system identification.

$$FPE = V\frac{1 + n/N}{1 - n/N}, \tag{33}$$

where V is the identification loss function, n is the number of estimated parameters, and N is the number of data pairs used in the estimation. The *FPE* criterion for higher model orders in the case of $n_a = n_b$ is illustrated in Fig. 10.

The above simulations showed that the proposed modeling approach provides appropriate performance in open-loop identification.

6.2. Closed-loop control system and the effect of controller parameters

In this part, the closed-loop characteristics of the AMP-TCP congestion control strategy are investigated and especially the effect of the control parameter, e.g., r, in the closed-loop system performance is studied. The set point is fixed to 90 ms and $r = 20$, 50, 100, 500, and 1000 are examined. Fig. 11 shows the congestion window size and the resulted averaged round-trip when r is set to 20 and 500.

Remark 8. Regardless of which r is picked, the network is stable and the system's output, the average round-trip time, reaches the desired set point of 90 ms. Increasing of r reduces input and output fluctuations around their desired and appropriate steady state values. However, it causes the system to be more sluggish (output reaches its desired set point slowly). This indicates a trade off between the faster response and tracking accuracy. Increasing of r, on the other hand, makes the system to be more stable as it can be deduced from (25) and (28). The observed fluctuations are basically because of the unpredictable randomness of CBR background traffic. If the CBR source is omitted, there would be no fluctuation.

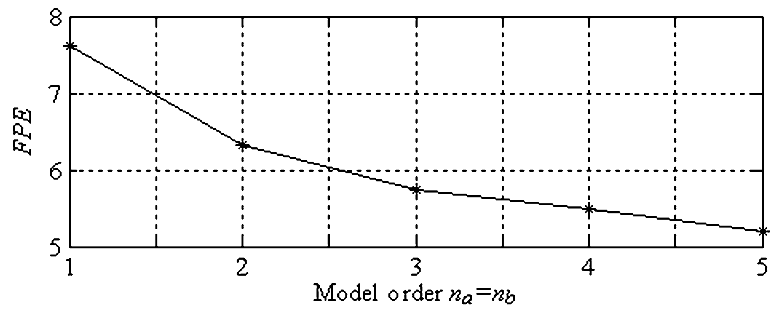

Fig. 10. *FPE* criterion for higher model orders in open-loop system identification for $\lambda = 0.995$.

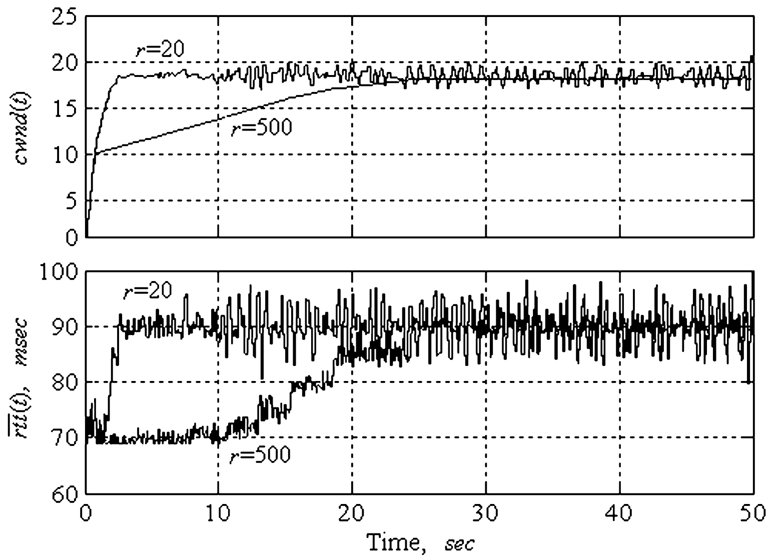

Fig. 11. Closed-loop simulation result using two different constant *r* values.

Remark 9. The second term in (16) is used to restrict the Δu or consequently the congestion window size *deviations*. As illustrated in Fig. 12, Δu is confined within $(-0.8, +0.8)$.

Remark 10. The recursive estimated model parameters are illustrated in Fig. 13. It is shown that the closed-loop system identification resulted in different estimated parameters in comparison with the open-loop identification as illustrated in Fig. 8. In fact, open-loop and closed-loop system identifications do not generally converge to the same estimations [15]. Closed-loop system identification performance should be basically evaluated based on the resulted closed-loop control performance.

When the system's output is far from the set point, rapid congestion window size change is required. On the other hand, when the response is near to its desired value, it is preferred to have less fluctuation on the variables. This goal can be achieved by adaptively adjusting the parameter *r* in accordance to the difference between the system's output and its desired value. The following rules are adopted in the adjustment of the parameter *r*.

$$r(t) = \begin{cases} 20 & |\Delta y(t) - \Delta y_d(t)| > 10, \\ 100 & 10 > |\Delta y(t) - \Delta y_d(t)| > 5, \\ 500 & 5 > |\Delta y(t) - \Delta y_d(t)|. \end{cases} \tag{34}$$

Use of the above rule improves the control performance as shown in Fig. 14. The system response is faster than using constant $r = 500$ and the steady state fluctuations are less than those obtained for $r = 20$. In the rest of the paper, *r* is always adjusted adaptively.

6.3. The effect of higher plant model orders

As pointed out in Section 6.1, the open-loop system identification could be improved using higher ARX model orders. In this part the same study is carried out on the closed-loop system. To do so, the control signal of (19) is modified using higher model's orders. Assume that the model is in the general ARX form of (35).

$$\begin{aligned} &\Delta y(t+1) + a_1 \Delta y(t) + \cdots + a_{n_a} \Delta y(t - n_a + 1) \\ &\quad = b_1 \Delta u(t) + \cdots + b_{n_b} \Delta u(t - n_b + 1) + e(t). \end{aligned} \tag{35}$$

Then the control cost function of (16) is modified as follows:

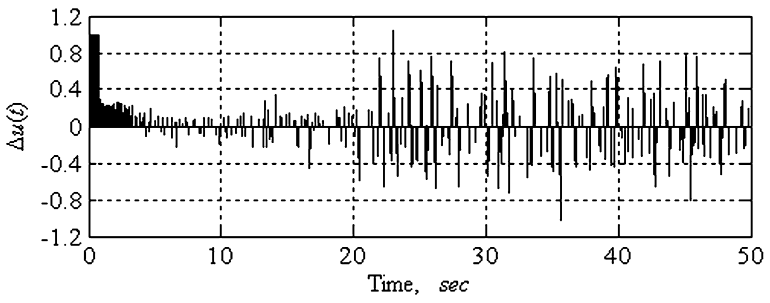

Fig. 12. Δu or the congestion window size deviations ($r = 100$).

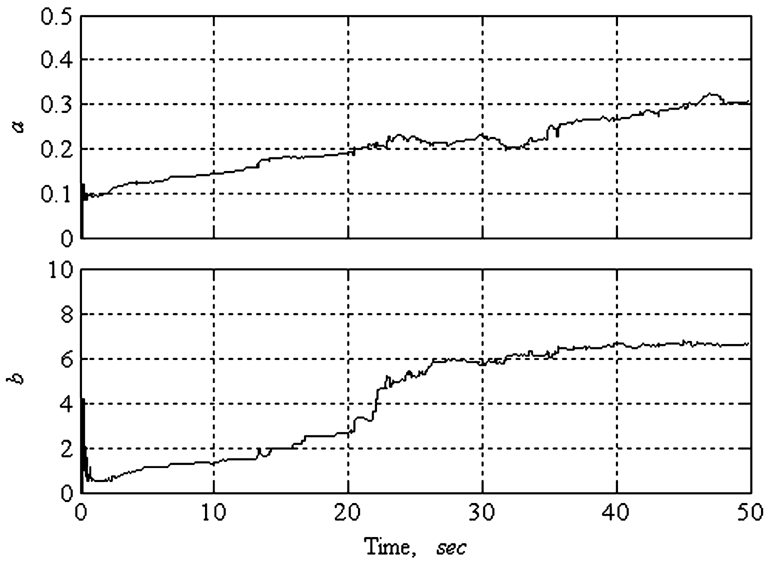

Fig. 13. Estimated model parameters during closed-loop system identification.

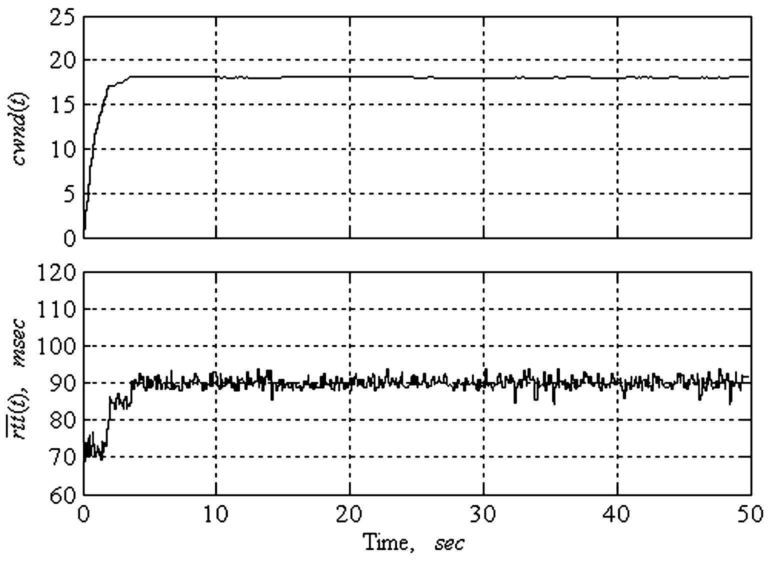

Fig. 14. Closed-loop simulation result using adaptive r adjustment.

$$J = (\Delta y_d(t+1) - a_1\Delta y(t) - \cdots - a_{n_a}\Delta y(t-n_a+1) + b_1\Delta u(t) + \cdots + b_{n_b}\Delta u(t-n_a+1))^2 + r\Delta u(t)^2. \quad (36)$$

The Δu that minimizes the modified control cost function is given by

$$\Delta u(t) = b_1\left(\frac{[a_1\Delta y(t) - d(t+1) + \Delta y_d(t+1)]}{r+b_1^2} + \frac{[a_2\Delta y(t-1) + \cdots + a_{n_a}\Delta y(t-n_a+1) - b_2\Delta u(t-1) - \cdots - b_{n_b}\Delta u(t-n_b+1)]}{r+b_1^2}\right). \quad (37)$$

The first term in (37) is the same as in (19). The second term is a correction that is caused by the model orders increase. To investigate the effect of the correction term, the closed-loop system is simulated for 50 s using higher model orders of the form of $n_a = n_b$ and the resulted systems are compared based on *tracking error criterion* which is defined as follows:

$$V_T = \frac{1}{N}\sum_{k=1}^{N}(\overline{rtt}_d(k) - \overline{rtt}(k))^2, \quad (38)$$

where N is the number of sampled data and $\overline{rtt}(k)$ is the k^{th} sampled averaged round-trip time in ms. The comparison results are illustrated in Fig. 15. The congestion window size and the averaged round-trip time when $n_a = n_b = 3$ and $n_a = n_b = 5$ are also illustrated in Figs. 16 and 17 respectively.

Remark 11. It is shown that by increasing the model orders, especially from the case of $n_a = n_b = 1$ to $n_a = n_b = 2$, the set point tracking is improved significantly. This increase speeds up the transient response although makes an overshoot in the system's output. The overshoot is decreased by further increase in the model orders. The improved control performance basically comes from the more accurate modeling and model-based prediction.

6.4. Ability to track different set points

In this part, the closed-loop system is run to track time variant set points as in (39). The simulation result is illustrated in Fig. 18. It is observed that the variable set point is tracked properly.

$$\overline{rtt}_d(t) = \begin{cases} 90 \text{ ms}, & 0 \leqslant t < 30, \\ 100 \text{ ms}, & 30 \leqslant t < 60, \\ 80 \text{ ms}, & 60 \leqslant t < 100. \end{cases} \quad (39)$$

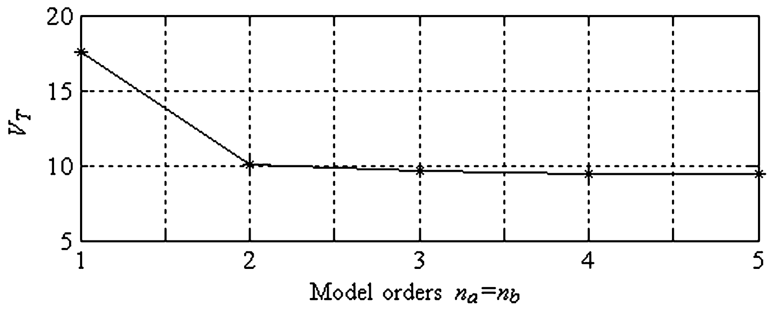

Fig. 15. Tracking error (V_T) for different model orders.

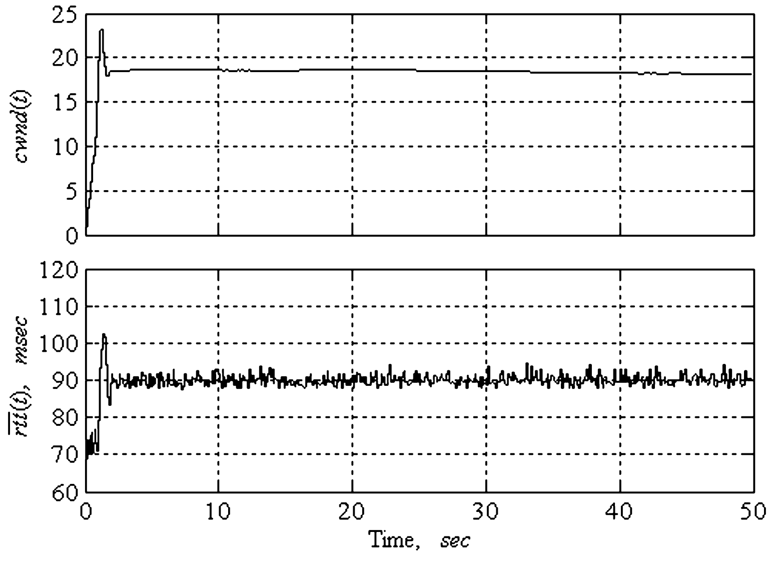

Fig. 16. Closed-loop simulation result for $n_a = n_b = 3$.

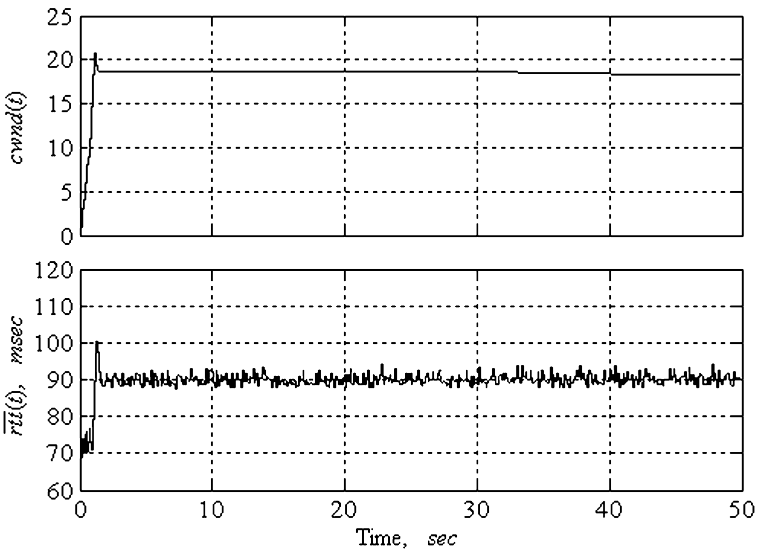

Fig. 17. Closed-loop simulation result for $n_a = n_b = 5$.

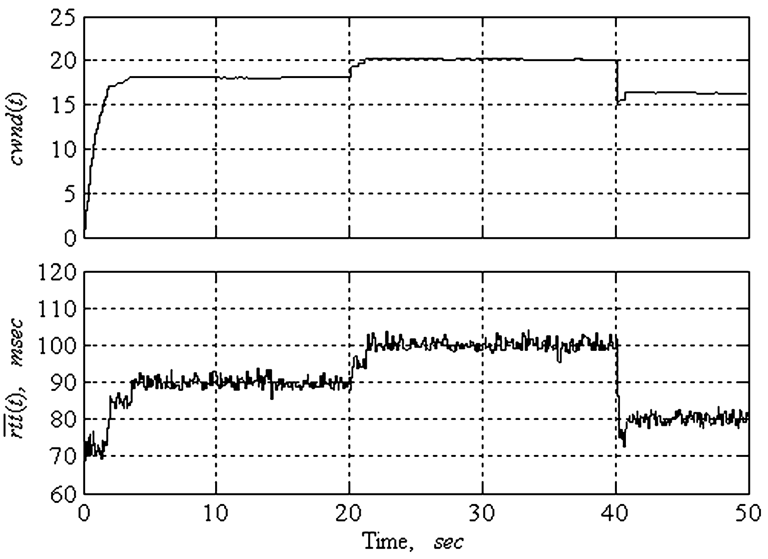

Fig. 18. Closed-loop simulation result when time variant set point.

6.5. Vegas-Like AMP-TCP in multiple-TCP scenario

In this part, a multi-source single-node network is considered. The network includes three Vegas-Like AMP-TCP sources and one CBR source. The desired set point in Vegas-Like AMP-TCP is $q_d = 2$. It is actually the average of TCP Vegas typical parameters of $\alpha = 1$ and $\beta = 3$. The other simulation parameters are the same as the previous simulations. In the simulated scenario, the first two sources submit their packets continuously during 20 ms of the simulations. The third source is connected and starts its packet submission at 10th simulation second. The estimated number of queued packets, adjusted congestion window size and also the averaged round-trip time is illustrated in Figs. 19–21 for each three AMP-TCP sources respectively. In all the three AMP-TCP sources, the system is stable and the number of queued packets tracks the desired $q_d = 2$. In addition, at 10th second, when the network dynamics is changed, the other two sources were able to adapt to the new network conditions and adjust their window size in such a way that the queued numbers of packets still track the desired value of 2. The automatically adjusted $\overline{rtt}_d$ in the three Vegas-Like AMP-TCP sources is illustrated in Fig. 22.

6.6. Comparing Vegas-Like AMP-TCP and TCP Vegas

To compare the proposed Vegas-Like AMP-TCP and the TCP Vegas, in terms of both transient and steady state behavior, the previous simulation repeated by using three TCP Vegas sources, with $\alpha = \beta = 2$. The simulation results are illustrated in Figs. 23–25.

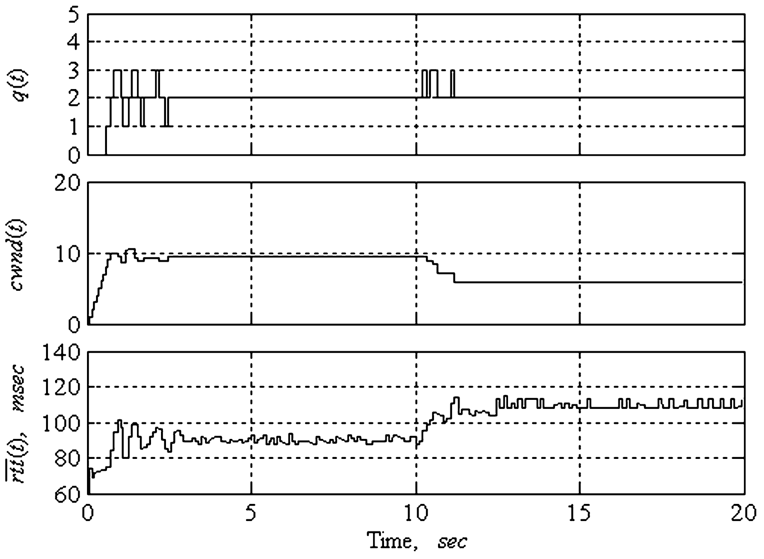

Fig. 19. Simulated result of Vegas-Like AMP-TCP #1 for multiple-source topology.

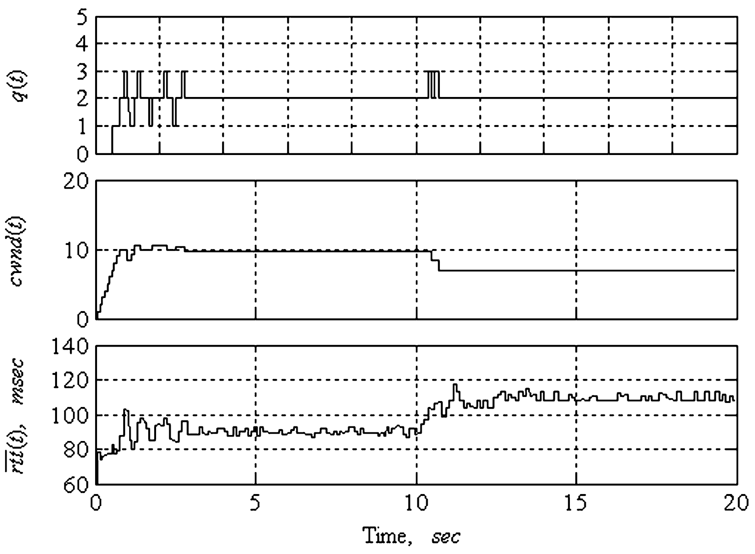

Fig. 20. Simulated result of Vegas-Like AMP-TCP #2 for multiple-source topology.

Remark 12. Considering the number of queued packets and the averaged round-trip time as the important network system's responses, it is observed that the AMP-TCP controller improves the *transient response* significantly in terms of the *percent overshoot* and the *settling time*. The improvement is considerable for all the three sources.

Remark 13. Considering the system responses after 10th simulation second, it is observed that the AMP-TCP controller is more capable to adapt with the new network conditions.

In summery, the Vegas-Like AMP-TCP is more successful than TCP Vegas to reach the aim of controlling the number of queued in-flight packets. It specially improves the transient behavior of the Vegas congestion avoidance phase.

7. Conclusion

Adaptive Model Predictive Transmission Control Protocol (AMP-TCP) as a new TCP delay-based congestion control algorithm is developed. A novel delay-based closed-loop congestion control problem was defined using systems and control theory. Then three research questions were outlined to solve the defined problem. To answer them: (1) Prediction error recursive system identification algorithm was used to model the network's delay dynamic. (2) Model-predictive control theory and adaptive control theory were employed to design an appropriate adaptive model-predictive congestion controller in TCP, called AMP-TCP. (3) The idea behind the TCP-Vegas congestion avoidance phase was followed to modify the AMP-TCP

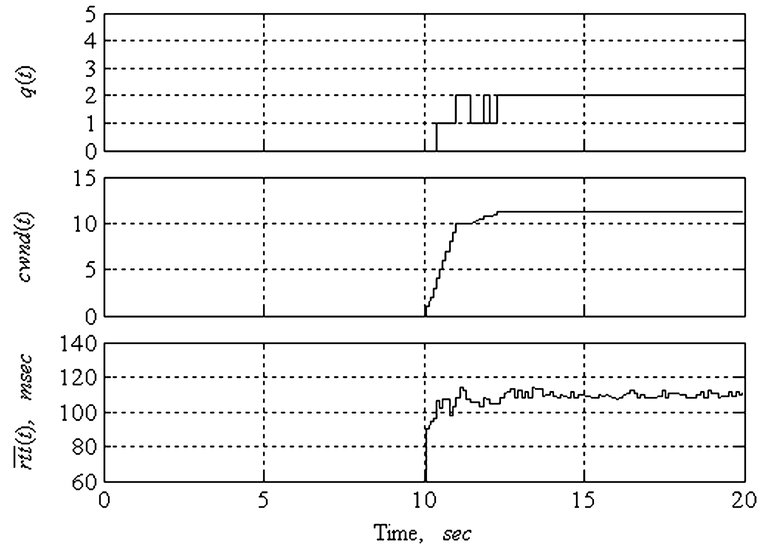

Fig. 21. Simulated result of Vegas-Like AMP-TCP #3 for multiple-source topology.

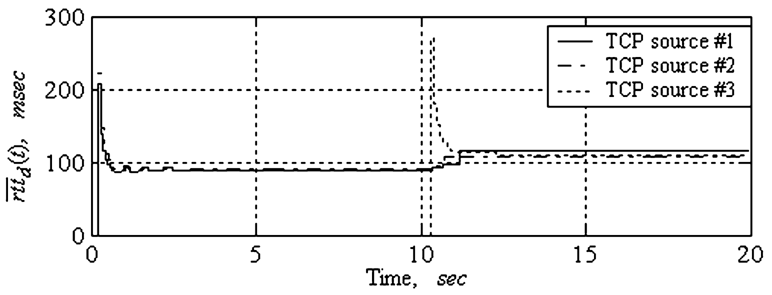

Fig. 22. Automatically adjusted desired RTT for each Vegas-Like AMP-TCP source.

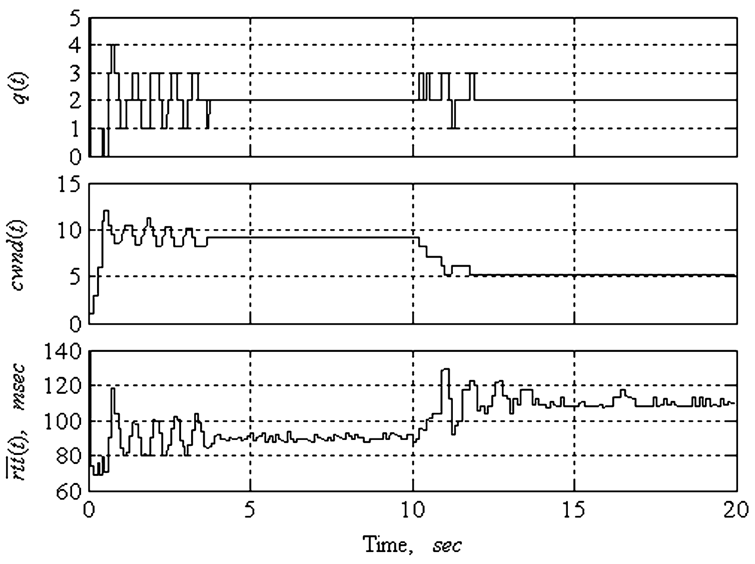

Fig. 23. Simulated result of TCP Vegas #1 for multiple-source topology.

controller and make it practical as a congestion controller in real computer networks with several sources. The modified protocol was called Vegas-Like AMP-TCP. The achieved performance by AMP-TCP and Vegas-Like AMP-TCP were studied via extensive *ns*-2 simulations.

The current study can be extended in several ways such as: (1) Adjusting the control parameter *r* as a continuous

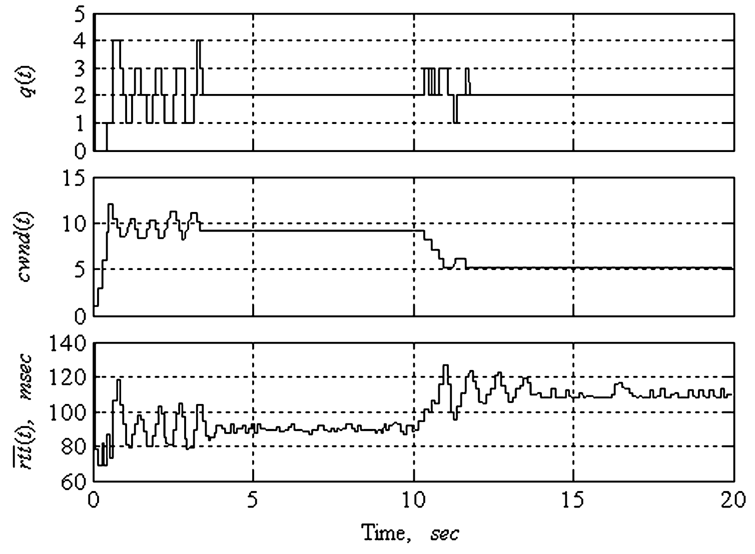

Fig. 24. Simulated result of TCP Vegas #2 for multiple-source topology.

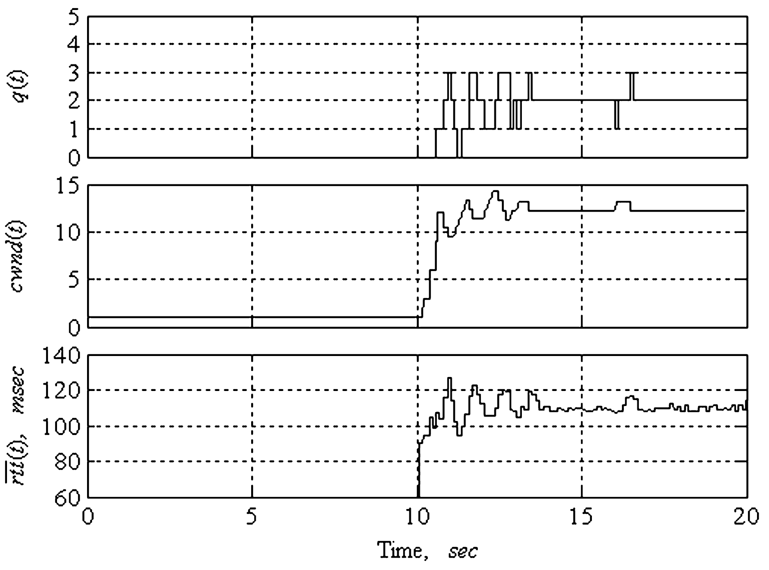

Fig. 25. Simulated result of TCP Vegas #3 for multiple-source topology.

function of tracking error. (2) Using different approaches to adjust the desired set point, instead of using number of queued packets. (3) Using more complex model structures, e.g., ARMAX and Box–Jenkins [14], in the modeling phase. (4) Increasing the prediction and the control horizons in the model predictive control design. (5) Evaluating the AMP-TCP and Vegas-Like AMP-TCP performance in more realistic and complicated simulation scenarios.

Acknowledgements

Financial support for the present work was granted by the Iranian Telecommunication Research Center (ITRC) under Grant number T500-8827. The authors, hereby, gratefully acknowledge this support.

References

[1] S.H. Low, F. Paganini, J.C. Doyle, Internet congestion control, IEEE Control Systems Magazine (2002) 28–43.
[2] C. Jin, D. Wei, S.H. Low, A case for delay-based congestion control, IEEE Computer and Communication Workshop, (2003).
[3] L.S. Brakmo, L.L. Peterson, TCP Vegas: end to end congestion avoidance on a global internet, Journal on Selected Areas in Communication 13 (8) (1995) 1465–1480.
[4] J.S. Ahn, P.B. Danzig, Z. Liu, L. Yan, Evaluation of TCP Vegas: emulation and experiment, in: Proceedings of SIGCOMM'95, 1995.
[5] E.F. Camacho, C. Bordons, Model Predictive Control, second ed., Springer-Verlag Publishing, London, 1995.
[6] The network simulator *ns*-2, <http://www.isu.edu/nsnam/ns/ urlhttp://www.isu.edu/nsnam/ns/>.
[7] V. Paxon, End-to-end internet packet dynamics, IEEE/ACM Transactions on Networking 7 (3) (1999) 277–292.

[8] P.F. Quet, H. Ozbay, On the design of AQM supporting TCP flows using robust control theory, IEEE Transactions on Automatic Control 49 (6) (2004) 1031–1036.
[9] Peng Yan, Yuan Gao, H. Ozbay, A variable structure control approach to active queue management for TCP with ECN, IEEE Transactions on Control Systems Technology 14 (2) (2005) 203–215.
[10] C.V. Hollot, V. Misra, D. Towsley, W. Gong, Analysis and design of controllers for AQM routers supporting TCP flows, IEEE Transactions on Automatic Control 47 (6) (2002).
[11] S. Deb, R. Srikant, Global stability of congestion controller for the internet, IEEE Transactions on Automatic Control 48 (6) (2003) 1055–1060.
[12] M. Haeri, A.H. Mohsenian Rad, TCP retransmission timer adjustment mechanism using system identification, in: Proceedings IEEE American Control Conference, Boston, June 30-July2, (2004), pp. 2328–2332.
[13] M. Haeri, A.H. Mohsenian Rad, TCP retransmission timer adjustment mechanism using model-based RTT predictor, in: Proceedings Asian Control Conference, Melbourne, 2004.
[14] L. Ljung, System Identification: Theory for the User, second ed., Prentice Hall, Upper Saddle River, NJ, 1999.
[15] K.J. Astrom, B. Wittenmark, Adaptive Control, second ed., Addison-Wesley Longman Publishing, Boston, 1994.
[16] D.P. Bertsekas, Nonlinear Programming, second ed., Athena Science, 2004.
[17] S.H. Low, L.L. Peterson, L. Wang, Understanding TCP Vegas: a duality model, Journal of ACM 49 (2) (2002) 207–235.
[18] Cheng.-Y. Ho, Chen.-Hua. Shih, Yaw.-Chung Chen, Yi.-Cheng. Chan, An Aided Congestion Avoidance Mechanism for TCP Vegas, Lecture Notes in Computer Science, vol. 3619/2005, pp. 961–971.
[19] C. Boutremans, J.Y. Le Boudec, A note on the fairness of TCP Vegas, in: Proceeding of International Zurich Seminar on Broadband Communications, 2000.
[20] Yi-Cheng Chan; Chia-Tai Chan; Yaw-Chung Chen, Performance improvement of TCP Vegas over heterogeneous networks, in: Proceeding of 24th International Conference on Distributed Computing Systems, 2004.
[21] Yi-Cheng Chan, Chia-Tai Chan, Yaw-Chung Chen, Cheng-Yuan Ho, Performance Improvement of Congestion Avoidance Mechanism for TCP Vegas, in: Proceeding of the 10th IEEE International Conference on Parallel and Distributed Systems, 2004.
[22] S. Vanichpun, Wu-chun Feng, On the transient behavior of TCP Vegas, in: Proceeding of the 11th International Conference on Computer Communications and Networks, (2002).
[23] Cheng-Yuan Ho, Yi-Cheng Chan, Yaw-Chung Chen, An Enhanced Slow-Start Mechanism for TCP Vegas, in Proceedings of the 11th IEEE International Conference on Parallel and Distributed Systems, 2005.
[24] A.H. Mohsenian Rad, Modeling and Control of Congestion in Computer Networks, M.Sc. Thesis, Department of Electrical Engineering, Sharif University of Technology, Iran, 2004.

Dr. Mohammad Haeri received the B.Sc. and M.Sc. degrees both in Electrical Engineering from Amirkabir University of Technology in 1985 and 1988 respectively. He got his Ph.D. in Electrical Engineering from University of Saskatchewan, Canada in 1994. Since 1994 he is with Electrical Engineering Department, Sharif University of Technology where he is now associate professor and head of the Advanced Control System Lab. His research area includes theory and application of model predictive control, computer networks modeling and control, control and synchronization of chaotic system, modeling of physiological systems. He is member of IEEE.

Amir-Hamed Mohsenian-Rad is a Ph.D. candidate in Department of Electrical and Computer Engineering at University of British Columbia. He received his B.Sc. from Amirkabir University of Technology (2002) and his M.Sc. from Sharif University of Technology (2004), both in Electrical Engineering. His current research is in the area of cross-layer optimization and resource allocation in mesh as well as mobile wireless ad-hoc networks, with past research in the area of congestion control and modeling in the Internet. He is an IEEE student member.