The CSI Journal on
Computer Science and Engineering
Vol. 5, No. 2&4 (b), 2007
Pages 56-68
Regular Paper

A New Algorithm for Guaranteed Delay in DiffServ Architecture

Mehran Mahramian Hassan Taheri Masoud Shafiee

Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran

Abstract

QoS support is necessary for the Internet new applications to meet end-to-end requirements such as bandwidth, delay, jitter and packet loss. Unfortunately, DiffServ architecture can only provide limited service assurance. Thus, a trade-off between simplicity of the implementation and strength of the service guarantees has to be made. In this paper, we have focused on DiffServ architecture to improve strength of service guarantees. We propose a new algorithm, using Adaptive Model Predictive Control as the controller to schedule differentiated buffers in routers. Simulation results show that the new algorithm can be used as a solution for the scheduling problem to guarantee QoS parameters like absolute delay, proportional delay and jitter.

Keywords: Quality of Service, Differentiated Services, Scheduling, Adaptive Control, Model Predictive Control.

1. Introduction

In the past few decades, the Internet has moved from a limited low bandwidth network to a sophisticated infrastructure supporting many new applications. Support of QoS in such a packet switched network requires a broad range of functions such as priority mechanisms, scheduling disciplines, traffic shaping, QoS signaling and routing algorithms. Several architectures such as DiffServ [1], IntServ [2], [3] and MPLS [4] have been proposed to solve QoS problems. Parameters such as scalability, strength of guarantee, and complexity show optimality of the proposed architectures, e.g. DiffServ has sacrificed strength of the service guarantee to improve scalability of IntServ architecture. We propose a new algorithm to make a trade off between IntServ accuracy and DiffServ scalability. By improving the service quality for packets in high priority queues, we would be able to guarantee delay for these services. In some previous works, the nonlinear queue model is linearized around an equilibrium point while classical control theory is used to check the stability of the algorithm as an offline tool [5]. In this paper, we employ the modern control theory to control the system using an adaptive linear model. Thus, we are able to check validity of the model as time goes on and adapt the model and the controller with nonlinearities and measured disturbances.

The proposed model is a multi input/multi output (MIMO) adaptive one, which is used by a model predictive controller to adjust the proportional delay of some classes of traffic in a broadband network. Unfortunately, classical control theory is difficult to apply for MIMO systems. The power of model predictive control has its roots in the fact that the state space model can represent a MIMO system as well as a SISO system. The rest of the paper is organized as follows. Previous works are introduced in Section 2. In Section 3, a brief explanation of adaptive model predictive control has been presented. Section 4 presents the details of system modeling. The Adaptive Model Predictive Control Scheduler (AMPCS) algorithm is presented in Section 5. Simulation results are shown in Section 6. Section 7 compares the results of the proposed algorithm with JoBS as one of the resent scheduling and buffer management algorithms. Finally, Section 8 concludes the paper.

2. Previous Work

Starting point of the DiffServ architecture was premium service [6]. It tries to simulate a leased line for the Virtual

Leased Line (VLL) users and assumes traffic shaping at the edge routers, so the VLL traffic does not exceed the contracted peak rate. It passes the VLL traffic as the highest priority traffic through the network. Premium service also assumes that this traffic is not bursty due to shaping at the edge routers, an assumption which is proven not to be true as a result of multiplexing of VLL traffic from different interfaces [7], [8].

The second strategy to implement the DiffServ architecture was assured service [9]. Each user may send traffic with a rate more than a contracted peak rate, which is signed as OUT traffic. In congestion conditions, The OUT traffic is lost with a significantly higher probability than legal traffic. Thus, users may send traffic more than the negotiated threshold in low traffic conditions, while they are limited to the threshold when congestion occurs. Recent experiments show that it is difficult to manage contracted traffic thresholds, while providing good service quality and simultaneously high resource utilization [10].

Many recent papers try to strengthen DiffServ accuracy while reducing complexity of the network especially at core routers [5], [7], [10], [11]. They do not assume any traffic shaping. They also minimize or avoid any on line negotiations about traffic conditions between routers. Scheduling at the routers is the main concern of these algorithms. They try to satisfy special constraints for delay of some aggregated traffic classes.

Dovrolis *et al*. proposed an idea to increase QoS accuracy, while preventing complexity [7], [12], [13]. They assume proportional delay and loss guarantees between different traffic classes instead of absolute delay or loss probability for each class. This means that we can guarantee that delay of the higher priority service is not more than a constant fraction of the delay of the lower priority service besides absolute boundaries for delay. This can be done using efficient scheduling algorithms.

Scheduling algorithms can be placed in two groups. In the first group, service rate is dynamically changed due to the number of waiting packets in each queue [5], [13]. These methods are basically advanced versions of GPS [14]. In the second group, priorities of queues are adaptively modified to satisfy the limits. Some algorithms in this group are Waiting Time Priority [13], Mean Delay Proportional [15], and Hybrid Proportional Delay Scheduler [13]. Only a few of these algorithms consider proportional loss probability. One of these algorithms is Proportional Loss Rate Dropper [12], which involves the problem of time dependency between loss calculation and loss proportion. Athuralia *et al*. provide a solution to this problem in [15].

Some recent works consider joint queue management and scheduling as a single problem [5]. In [16], the RIO algorithm is considered as the superposition of two RED algorithms, which are applied to the in band and out band traffic. Recently, Liebherr *et al*. [17] propose a powerful algorithm named JoBS. The algorithm is used in the QoSBox, which is a machine guaranteeing QoS in backbone of DiffServ architecture. Striegel *et al*. [18] use a similar algorithm, but their main contribution is multicast. A new algorithm proposed in [19], named Equivalent Differentiated Services, uses the waiting time priority [13] and proportional loss [12] to control queue input and output, respectively.

Control theory is used in computer networks as a powerful tool to improve robustness, accuracy and throughput. Most researchers use classic and linear control tools [20], [21]. However, some of them use control theory to model their algorithm and find stability conditions [22], [23], [24], [5]. Although predictive control is used for congestion control algorithms [20], our search results show that no previous research has used model predictive control to solve scheduling problem.

In our previous work [25], we proposed MPCS algorithm, which uses model predictive control to schedule the packets coming from different classes of traffic to guarantee proportional delay. Although MPCS algorithm is proved to be feasible, it suffers from lack of an accurate model. In this paper, we use adaptive model predictive control to overcome inaccuracy. Our simulation results show that we succeeded to reach our goal to propose an accurate model.

3. Adaptive Predictive Control Principles

Model predictive control, as one of the strongest methods of modern control theory, has developed considerably in the last decades. Clarke *et al*. presented a seminal idea in [27] and the industrial model predictive control technology is reviewed in [28].

The main attractions of MPC that motivated us to apply it to the scheduling problem were that it can intrinsically take account of system and controller constraints and naturally handles multivariable control problems. It also allows operation closer to the boundaries of constraints compared with conventional control. Besides, it handles an event based system as well as a time based system. The main ideas in model predictive control are [29]:

- It uses a model to predict output of the system in the future.

$$\begin{aligned} &y(n+1) = Ay(n) + Bu(n) \\ &y(n+2) = A^2 y(n) + ABu(n) + Bu(n+1) \\ &\vdots \\ &y(n+H_p) = A^{H_p} y(n) + A^{H_p-1} Bu(n) + ... + Bu(n+H_p-1) \end{aligned} \tag{1}$$

where H_p is the receding horizon. Receding is one of the brilliant ideas of MPC. It allows the optimization problem to be solved in the open loop, which reduces the complexity. It will be discussed in more detail in the following paragraphs.

- It calculates a control input to optimize an objective function. Thus, it can be considered as an optimal control method. The optimization algorithm uses the system model and an objective function containing the measured output and the desired output to calculate the optimum inputs in the present time and the future. The criterion to be minimized is as follows:

$$J(n) = \sum_{i=H_w}^{H_p} \left\| y(n+i) - s(n+i) \right\|_{W(i)}^2 + \sum_{i=0}^{H_u-1} \left\| \Delta u(n+i) \right\|_{R(i)}^2 \tag{2}$$

where $\|x\|_W^2 = x^T W x$, y is the predicted output of the system, s is the set point vector, $\Delta u(n)$ is the difference between manipulated variable u at the n^{th} and the $(n\text{-}1)^{\text{th}}$ steps, H_p is the receding horizon, H_w is the starting step for penalizing output deviation from the set point, H_u is the control horizon, i.e. number of steps that u is allowed to be

changed and $W(i)$ and $R(i)$ are positive weighting matrices to penalize deviations on some steps, e.g. deviations on ending steps of receding and control horizon may be penalized more intensely than deviations on the first steps. Then the optimization problem is:

Determine $u(n)$, $u(n+1)$, …, $u(n+H_p)$ subject to constraints, so that $J(n)$ is minimized. As a matter of fact, $J(n)$ penalizes output deviations and rapid changes of control activity. In our problem, rapid control activity may cause the output traffic to be much bursty.

- At each step, all calculations are repeated to optimize the objective function; however, calculations are repeated every N steps in some kinds of MPC [29]. This idea lets considering an open loop optimization problem, which is much easier to solve than a sophisticated closed loop one.
- One of the strengths of MPC is that it explicitly considers constraints. If the constraints are in the form:

$$\begin{aligned} E\begin{bmatrix}\Delta u(n)\\ 1\end{bmatrix} &\le 0\\ F\begin{bmatrix}u(n)\\ 1\end{bmatrix} &\le 0\\ G\begin{bmatrix}y(n)\\ 1\end{bmatrix} &\le 0 \end{aligned} \tag{3}$$

where E, F and G are matrices which are determined by constraints. Using Equation (1), all three inequalities in (3) can be rewritten as a function of $\Delta u(n)$:

$$\begin{bmatrix}E'\\ F'\\ G'\end{bmatrix}\Delta u(n) \le \begin{bmatrix}E''\\ F''\\ G''\end{bmatrix} \tag{4}$$

where E', F', G', E'', F'' and G'' are derived from inequality (3) and Equation (1). The Equation (2) can also be rewritten in the form:

$$J(n) = \Delta u(n)^T H \Delta u(n) - P\Delta u(n) \tag{5}$$

Equation (5) and the inequality (4) are in the standard form of Quadratic Programming (QP). Fortunately, this kind of optimization is convex [30]. The convexity guarantees termination of the optimization problem. It also guarantees that a global minimum is eventually reached and there is no local minimum.

Most of the time, the main problem for applying a model predictive controller to a system is finding an accurate model. The model accuracy has a direct relation with the performance of the model predictive controller. As a matter of fact, the controller predicts the behavior of the system based on the model. If the model is not accurate, the predictions will have errors and this makes the performance of the controller poor.

As the scheduling system is nonlinear, we tried to use an adaptive model to overcome the problem. Adaptive control has been used for a long time to control high dynamic nonlinear systems [31]. In this paper, the Model Predictive Controller (MPC) is employed that uses a model to predict the system behavior. Using the adaptive scheme, the model and the controller are improved as time goes on. The main advantage of adaptive control is that it omits the need for complex analytical models while coping with high dynamism of the system and providing relatively accurate local models. Figure 1 shows an adaptive control scheme. The proposed algorithm, uses indirect adaptive MPC to update model parameters in each sample time and determine control signal in accordance. Next session describes the model.

4. System Modeling

Consider a router in which packets are arriving from input ports and departing from output ports. There are some blocks in a router such as packet classifier, IP lookup, buffer manager and scheduler (Figure 2). The proposed algorithm deals with the scheduler. Packets entering the router have labels indicating the class of traffic they belong to. The classifier places the packets in the related queues. The packets must wait in the queues until the scheduler decides to serve them. The system contains Q queues and packets with high priority queues should encounter less delay than those in the low priority classes. Note that while the actual input and output of the queue are packets, from systematic point of view, the service rate and the delay are considered as the input and output of the queue model respectively (Figure 3). For each queue, the scheduler adapts the service rate, in a way that delay of the related queue gets close to the guaranteed value. Increasing the service rate decreases the delay. The main constraint is that the system is limited to the output bandwidth capacity. It should also be work conservative i.e. the scheduler should use the maximum available bandwidth. Therefore, the sum of the manipulated variables or service rates should be equal to the output bandwidth of the system.

As described earlier, our aim is to use MPC to provide proportional differentiation on average delay for different traffic classes. Let us assume that there are Q classes in the system. Let the measured average output of class i at event n be $\bar{y}_i(n)$. In a proportional delay service, the proportion of measured outputs between classes i and $i+1$ is assumed to be some predetermined value k_i:

$$\frac{\bar{y}_{i+1}(n)}{\bar{y}_i(n)} = k_i, \quad i = 1, \ldots, Q-1 \tag{6}$$

The network administrator decides the value of k_i. In order to provide proportional delay guarantee, a controller is applied to the system. Figure 4 shows the block diagram of the system.

The object of the controller is to modify the service rates in a way that the proportional delay error is minimized. From the computational point of view, the proportion of two parameters is not a proper objective function for the controller, so a new parameter is introduced, which is the sum of the weighted delays of the queues:

$$y(n) = \frac{\sum_{i=1}^{Q}(\prod_{k=1,k\neq i}^{Q} m_k) y_i(n)}{Q} \tag{7}$$

where $m_i = \prod_{j=1}^{i-1} k_j$, and $m_1 = 1$. Theorem 1 shows that the new parameter $y(n)$ can be used as the set point of the system.

Theorem 1: *If the weighted average output of each class:*

$$y_i^*(n) = (\prod_{k=1,\, k\neq i}^{Q} m_k) y_i(n), \quad 1 \le i \le Q \tag{8}$$

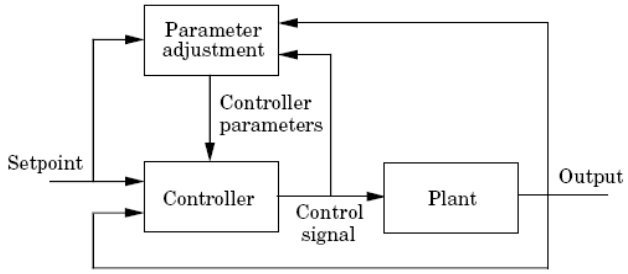

Figure 1. Block diagram of an adaptive control system

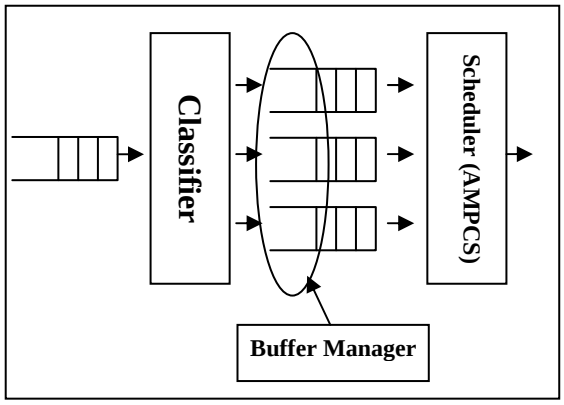

Figure 2. Basic parts of a router in DiffServ architecture

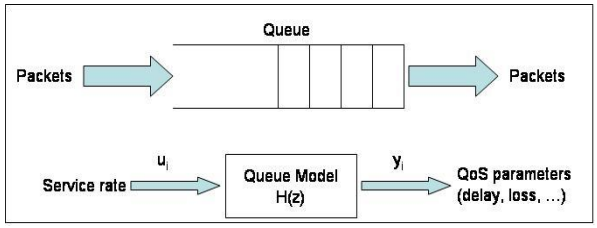

Figure 3. A queue and a queue model. The queue (up) used in the queue theory. The queue model (down) used for MPC design

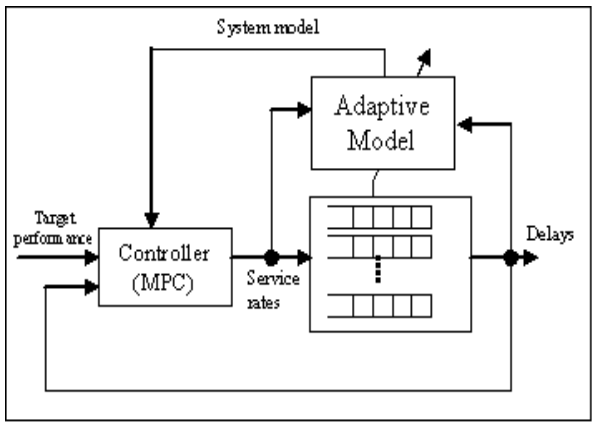

Figure 4. Block diagram of AMPCS

goes toward $y(n)$ *, which is defined in (7), then the proportional delays get close to the desired values.*

Proof: The controller in any system tries to adjust the manipulated variables in order to minimize the difference between output variables and predetermined set points. If $y(n)$ is assumed as the set point and $y_i^*(n)$'s as the outputs of the system, all output $y_i^*(n)$ should converge to $y(n)$. Then the steady state can be written as the following:

$$y_{ss} = y_{i,ss}^*, \quad \forall i \in \{1, 2, \cdots, Q\} \tag{9}$$

where y_{ss} and $y_{i,ss}^*$ are steady state values of $y(n)$ and $y_i^*(n)$, respectively. Combining (8) and (9) gives:

$$\frac{y_{i+1,ss}}{y_{i,ss}} = \frac{\frac{y_{ss}}{\prod_{p=1,p\neq i+1}^{Q} m_p}}{\frac{y_{ss}}{\prod_{p=1,p\neq i}^{Q} m_p}} = \frac{\prod_{p=1,p\neq i}^{Q} m_p}{\prod_{p=1,p\neq i+1}^{Q} m_p} = \frac{m_{i+1}}{m_i} = \frac{\prod_{j=1}^{i} k_j}{\prod_{j=1}^{i-1} k_j} = k_i \tag{10}$$

Therefore, if the $y(n)$ is considered as the set point and the $y_i^*(n)$ as the output of the system, it gives the same results if k_i's are set points and the $\bar{y}_{i+1}(n)/\bar{y}_i(n)$'s are the output variables.

Thus, a model is needed to describe the interaction between service rates (as the inputs of the system) and delays (as the outputs of the system). An adaptive scheme is used to explore the model in each step. Due to high level of dynamism in the network, the model parameters are affected by several noise and disturbances such as input traffic and network congestion. Ying Lu *et al.* in one of their recent works used the adaptive model for the same problem in a web server [11]. They assumed that parameters are constant but unknown. The experience shows that parameters are strongly affected by traffic conditions. In this paper, existing disturbances and noises are considered as white measurement noise in the model.

The queue dynamics is strongly nonlinear. To approximate the system model, an adaptive queue model based on an Auto Regressive eXogenous (ARX) structure is utilized. The disadvantage of using such a model is that the system itself is strongly nonlinear and the ARX model is a time invariant linear model. Adaptive modeling helps us to overcome the problem, because it can adapt the model parameters to follow the dynamics of the system. In other words, a linear time variant model structure is employed to describe behavior of the nonlinear system.

$$H_i(z) = \frac{Y_i(z)}{U_i(z)} = \frac{\sum_{j=0}^{m} b_{ij} z^{-j}}{1 + \sum_{j=1}^{p} a_{ij} z^{-j}} \tag{11}$$

$Y_i(z)$ and $U_i(z)$ are z transform of $y_i(n)$ and $u_i(n)$ and p and m are number of poles and zeros of the model. The relation between $y_i(n)$ and $u_i(n)$ in time domain is given as:

$$y_i(n) = -a_{i1}y_i(n-1) - \cdots - a_{ip}y_i(n-p) + b_{i1}u_i(n-1) + \cdots + b_{im}u_i(n-m) + e(n) \quad (12)$$

where $e(n)$ represents the modeling error. This model is converted to the state space model:

$$y(n+1) = Ay(n) + Bu(n) + \omega(n) \quad (13)$$

where $y(n)$ and $u(n)$ are Q-dimensional vectors of outputs and inputs. A and B are system dynamic matrices and $\omega(n)$ is the modeling error. The state space model is used in AMPCS algorithm to predict the future state of the system. To determine the model parameters, the recursive least squares estimation is implemented:

$$y_i(n) = \varphi_i^T(n-1)\theta_i \quad (14)$$

where

$$\varphi_i^T(n-1) = \left[-y_i(n-1), \cdots, -y_i(n-p), u_i(n-1), \cdots, u_i(n-m)\right] \quad (15)$$

$$\theta_i^T = \left[a_{i1}, \cdots, a_{ip}, b_{i1}, \cdots, b_{im}\right] \quad (16)$$

The recursive least squares algorithm is defined by the following equations:

$$\hat{\theta}_i(n) = \hat{\theta}_i(n-1) + P_i(n)\varphi_i(n-1)(D_i^*(n) - \varphi_i^T(n-1)\hat{\theta}_i(n-1)) \quad (17)$$

$$P_i(n) = \lambda^{-1}(P_i(n-1) - P_i(n-1)\varphi_i(n-1)\varphi_i^T(n-1)P_i(n-1)(\lambda + \varphi_i^T(n-1)P_i(n-1)\varphi_i(n-1))^{-1}) \quad (18)$$

$\hat{\theta}_i(n)$ is the estimated value of $\theta_i(n)$ and $\lambda \in [0,1]$ is the forgetting factor.

The model predictive control uses the adaptive model that is described by (11) and the estimated parameters $\hat{\theta}_i(n)$ to control the system.

5. Adaptive Model Predictive Control Scheduler (AMPCS)

In this paper we apply an adaptive model predictive controller to the scheduling problem and investigate the results. We would like to show that adaptive MPC is able to control a scheduling system precisely and satisfy proportional delay constraints. We call the proposed algorithm Adaptive Model Predictive Control Scheduler. The first step to design a model predictive controller is the system modeling. The system here consists of Q queues with the first queue having the highest priority. Packets arrive in their corresponding queues and wait until being serviced. Thus, there are Q systems, which the manipulated variable of each system is the service rate and the output variable is the delay of the corresponding queue. The purpose of the controller is to regulate the service rates (manipulated variables) of different queues, so the proportional delays become equal to predetermined values, e.g. k_i. The proportional delay is guaranteed and no absolute guarantees are considered. In our previous work [25], a simple linear first degree model proposed in [5], was used. In this paper, we use adaptive modeling to increase the accuracy of the algorithm. In each step, the model of the system is calculated based on the past time behavior of the system. This model, which is updated in each step, is used to predict the future behavior of the system. The prediction, which is based on an adaptive model, is the main key idea of the adaptive model predictive control.

To explain our algorithm, we assume:

n: number of packet arrivals since start of the last busy period.

$D_i(n)$: Delay of the packet in the head of the i^{th} queue, i.e. the time since arrival of this packet.

$B_i(n)$: Total backlog packets in the i^{th} queue.

$r_i(n)$: Service rate of the i^{th} queue.

$\Delta r_i(n)$: The difference between service rate of previous and present time intervals.

k_i: The proportional guarantee on delay, i.e. the proportion of $\overline{D_{i+1}}$ to $\overline{D_i}$, which are average delays of the related queues.

C: The output link capacity.

It is obvious that $\Sigma r_i(n) \leq C$.

The desired value (set point) for the delay of each class should be calculated due to delay of other classes. To obtain a uniform object function, based on the average delays of different queues, a new weighted delay is defined:

$$D_i^*(n) = \left(\prod_{k=1,k\neq i}^{Q} m_k\right) D_i(n) \quad (19)$$

in which, $m_i = \prod_{j=1}^{i-1} k_j$ are temporary variables to define a uniform object function for all traffic classes. m_1 is assumed to be one.

The object function is to make all $D_i^*(n)$ s equal, i.e. change service rates of traffic classes, to minimize absolute differences of $D_i^*(n)$ s.

To estimate the model parameters, an adaptive queue model based on an Auto Regressive eXogenous (ARX) structure is used. We may assume the system is a black box and model it using an ARX model. The disadvantage of using such a model is that the system itself is strongly nonlinear and the ARX model is a time invariant linear system. Adaptive modeling overcomes this problem, because it models the system in a short interval in the past time and uses the model for a short interval in the present time and the future, before the dynamics of the system affect the model parameters considerably.

The equation (20) shows an ARX model i.e. the transfer function between $D_i^*(n)$ (output of the system) and $r_i(n)$ (service rate which is manipulated variable) in the specified model (Figure 3):

$$H_i = \frac{Z[D_i^*(n)]}{Z[r_i(n)]} = \frac{\sum_{j=0}^{m} b_j z^{-j}}{1 + \sum_{j=1}^{n} a_j z^{-j}} \quad (20)$$

m and n are number of zeros and poles of the system and b_i and a_i are time varying coefficients that are determined from a recursive least square algorithm with a forgetting factor to track parameter variations (Section 4).

Q traffic classes have Q transfer functions as described in equation (20). Thus, H is a multiple input multiple output (MIMO) transfer function, which is a diagonal Q by Q matrix.

$$H = diag(H_1, H_2, ..., H_Q) \tag{21}$$

This can be assumed as Q independent transfer functions, but there is a constraint on output link capacity, which relates these independent transfer functions:

$$\Sigma r_i(n) \leq C \tag{22}$$

We like the system to be "work conservative", i.e. the throughput of the output link be 1 while there are some packets waiting in any queue. Therefore, the above equation is modified to:

$$\Sigma r_i(n) = C \tag{23}$$

Thus, the system is Multiple Input Multiple Output (MIMO) and not simply a collection of some independent single input single output (SISO) systems. Fortunately, the model predictive controller can control a MIMO system as well as a SISO one. The ARX model in fact calculates the output at the next step ($D_i(t)$) using the previous inputs and outputs of the system. The RLS algorithm is used to estimate the system parameters (Section 4).

The optimization problem considered consists of a worst case quadratic performance criterion over a linearized discrete-time model subject to inequality constraints on the states and control signals. The constraints are (10) and:

$$L_i < r_i < C \tag{24}$$

L_i avoids starvation for the i^{th} queue and is the lower band for i^{th} service rate and C is the link capacity. One of the problems encountered during algorithm design was the work conservativeness of the controller. If we would like to have a work conservative system, the constraint (23) should be applied to the MPC.

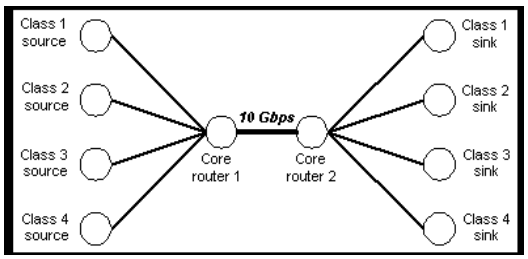

Figure 5. The network topology for the simulation. There are 2 core routers. The AMPCS scheduler is run at the core router #1

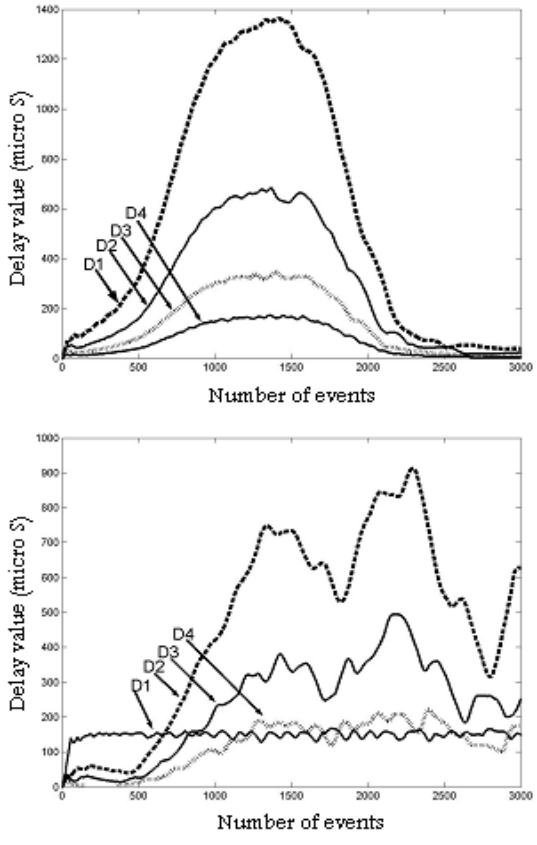

Figure 6. Delay values for two scenarios. proportional delay (up) and proportional and absolute delay (down) are controlled by our algorithm

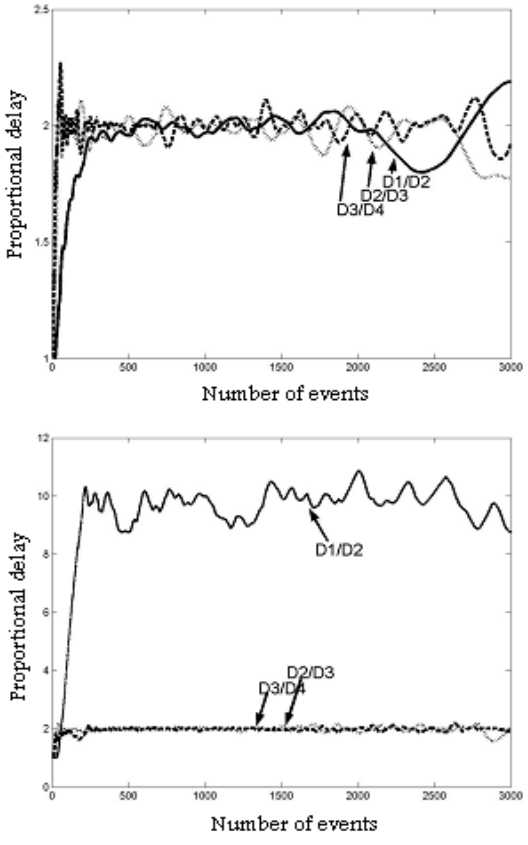

Figure 7. Proportional delay for the first configuration. Set points are equal (up). One of the set points is 10 (down)

6. Simulation Results

In the simulation, four service levels were assumed (Q = 4). The service rates were adjusted at each event. Input traffic is considered as 100 users that produce traffic with Pareto distribution and α equal to 1.2. The same rate for all classes was chosen. Therefore, simulation is done in the worst case. Packet sizes are assumed to have a uniform distribution between 50 and 650 with the same average for all classes. Simulation topology is shown in Figure 5.

At the first scenario, k is set to 2 for all classes i.e. the average delay of each class should be two times the average delay of the higher class. Buffers are assumed to be infinite (Figure 6, up). In Figure 7, proportional delays are shown.

All k_i's are equal to 2 in the Figure 7 (up). But, k_1 is set to 10 in the Figure 7 (down), assuming best effort service for the traffic of class 1.

At the second scenario, one of the classes has the absolute delay equal to 155 time units and k is set to 2 for the other three classes (Figure 6, down). One of the brilliant results of our algorithm, which can be seen in the results of the second configuration, is restricting jitter amount to a very low value. It can be seen in the Figure 6 (down) that although the system is work conservative, the jitter amount is very low.

The object function should satisfy two objectives:

- Adjust the proportional and absolute delays to the specified values, i.e. k_i or d_i.
- Avoid excessive changes to the service rates allocated to the classes.

To satisfy the first objective, the controller tries to change the service rates, so the absolute and proportional delays become as near as possible to the set points (d_1=155, k_i=2). However, the second one ensures that the controller does not change the service rates with large amounts, which reduces burstiness of the output traffic.

Tables 1 and 2 show the error of five different data sets applied to the controller for the first and second configuration.

To check the model validity, the topology in Figure 8 was utilized [33]. Figure 9 shows the delay predictions made by model and the real delay values during simulation. It can be seen that the predicted values are very close to the actual values. Figure 10 shows the normalized estimation error.

$$NEE = 10\log_{10}\frac{(D_i - \hat{D}_i)^2}{D_i^2} \tag{25}$$

where D_i is the real delay of the i'th class and $\hat{D}_i$ is the estimated value. To save the space, only NEE of the first class is shown in the figure.

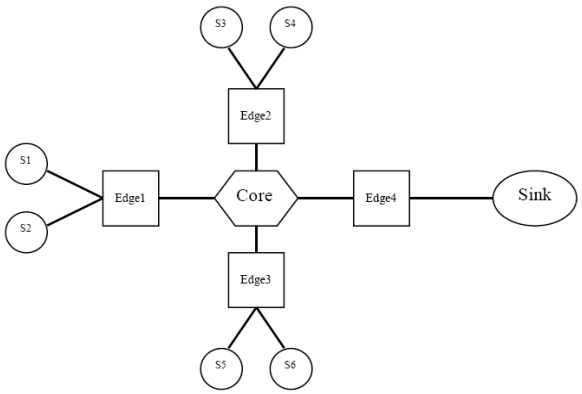

Figure 8. Simulation topology, which is used for checking model validity. Traffic sources 5 and 6 produce background traffic

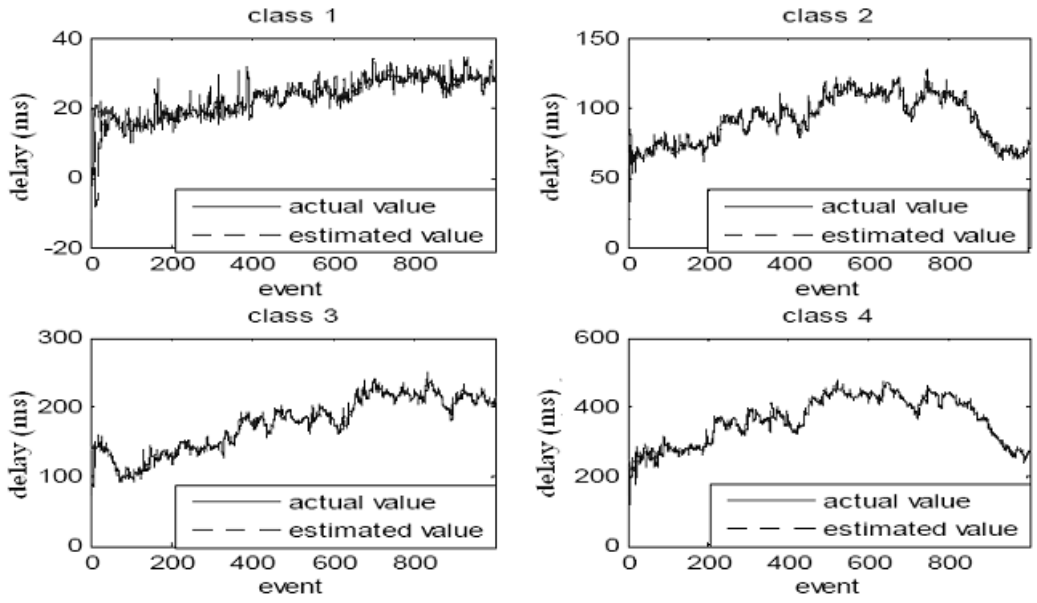

Figure 9. Predicted delay versus actual delay value

Also, in [26], it was shown that our previous model is not so accurate and the adaptive model is a perfect model. Thus, the adaptive model is accurate and satisfies the predictive control needs.

7. Comparing Our Algorithm (AMPCS) With JoBS

This chapter deals with comparing the simulation results of our algorithm (AMPCS) with JoBS [5], which is one of the newest and most accurate scheduling algorithms. JoBS allocates a guaranteed service rate to each class and dynamically shares the remaining bandwidth between classes in a way that proportional and absolute delay constraints are met. Although the JoBS algorithm calculates the service rates based on an optimization problem, they propose a heuristic algorithm to reduce the complexity. In the JoBS heuristic algorithm, the delay constraint errors are monitored every N arrival of packets. If the error is less than a threshold, the service rates do not change, otherwise, an adaptive proportional controller is used to change the service rates.

Our goals are:

1) Determine how well our algorithm can guarantee proportional delay in different situations and (2) Compare our algorithm with JoBS heuristic.

In this simulation, we use the following two schemes. Both the simulations are performed in NS-II [32] environment in the completely similar conditions.

AMPCS is configured as follows:

- ARX model with 1 pole and no zeros.
- H_p, the receding horizon of MPC, is 1.
- H_u, the control horizon of MPC, is 1.
- For the set points, an exponential curve was used with time constant equal to 10 events. Note that our system is event based.
- λ parameter of RLS algorithm, which determines convergence speed, is 0.99.

2) JoBS algorithm has some fixed parameters for configuration, as in [17]. The algorithm is now included in NS-II package. So, we have used the same code within simulations in [17].

There are also some common conditions:

- All the proportional delay values (k_i) are 2.
- 4 different levels of service are considered.
- The output link capacity is 10 Mbps.
- The memory capacity is shared between queues and is 500 packets.
- The network topology is the same as used in the previous simulation, i.e. there are two routers with a 10 Mbps link between them. There are also 4 users that produce two types of traffic. The first type is constant bit rate and the second type has Pareto distribution with α equal to 1.2 for each of the classes. All four classes and users have the same traffic rates i.e. each user produce 25% of the whole traffic.
- The algorithm for dropping packets due to buffer overflow is tail drop.

The complexity of the MPC is $O(\bar{n}^3)$, where $\bar{n}$ is the number of manipulated variables multiplied by control horizon i.e. $\bar{n} = H_u(Qp)$ [34]. Both number of poles and control horizon in this paper are 1. So, the complexity of the algorithm is $O(Q^3)$. Optimization is the dominant part in MPC algorithm. The optimization consists mainly of matrix operations that are matrix inversion and matrix multiplication.

Table 1. Percent error rates for the proportional delay in the first configuration.

D1/D2 error (%)	D2/D3 error (%)	D3/D4 error (%)
1.67	2.77	3.28
5.76	4.8	4.17
1.94	3.21	3.76
2.49	1.45	2.26
1.33	1.17	1.51

Table 2. Percent error rates for the proportional and absolute delay in the second configuration.

Absolute delay error (D1) (%)	D2/D3 error (%)	D3/D4 error (%)
19.92	6.65	4.95
21.37	5.87	4.03
25.68	7.84	3.57
18.01	5.12	6.42
19.06	5.90	2.96

As a matter of fact, the most important limitation of MPC is the computational load. The high computational complexity has restricted the use of this technology to relatively slow systems mainly encountered in the chemical and petrochemical industry. Fortunately, the number of traffic classes is most of the time a small value (e.g. 4) and the problem of computational complexity does not seem to be serious. However, many heuristic algorithms are developed to decrease the complexity of the MPC implementation [34]-[37].

The methods that are used to propose a low complexity MPC include input trajectory parameterization, developing a search tree to find the optimum values [35], proposing a new architecture for a special purpose system on a chip processor [34] and heuristic algorithms for approximation of solving quadratic programming optimization problem [37]. Tyagunov, in his Ph.D. dissertation [36], has developed a new routine for efficient calculation of solutions of this problem.

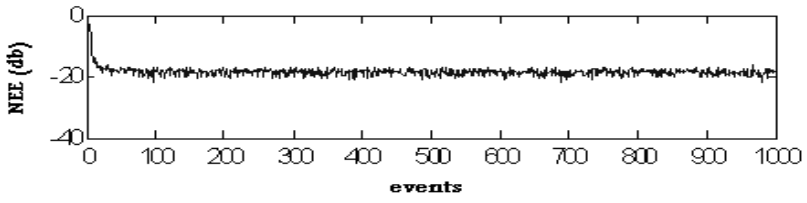

Figure 10. Normalized Estimation Error of the first class delay during simulation. The NEE of the other three classes are similar

Table 3. The following metrics evaluate the two algorithms for two types of traffic, constatnt bit rate and Pareto. The latter type is much bursty

Algorithm	Input traffic	Averaged prop. Ins. error $E_{inst,i}$ (%)			Average prop. delay error $\overline{E_i}$ (%)			Maximum delay $D_{Max,I}$ (μs)				Maximum prop. delay error $E_{Max,i}$ (%)		
		i=1	*I*=	*i*=3	*i*=1	*i*=2	*i*=3	*i*=1	*i*=2	*i*=3	*i*=4	*i*=1	*i*=2	*i*=3
AMPCS	CBR	24.0	12.4	7.8	1.65	1.31	1.90	8701	12975	25518	40653	2753	791	502
JoBS	CBR	158	146	75.2	299	130	60.8	5468	13403	32190	65844	3468	6572	16261
AMPCS	Pareto	29.5	28.8	29.8	79.5	74.9	49.1	6462	28560	29820	40456	1208	1188	3108
JoBS	Pareto	30.6	23.6	21.9	10.0	6.2	14.6	3894	7179	14862	34640	608	466	717

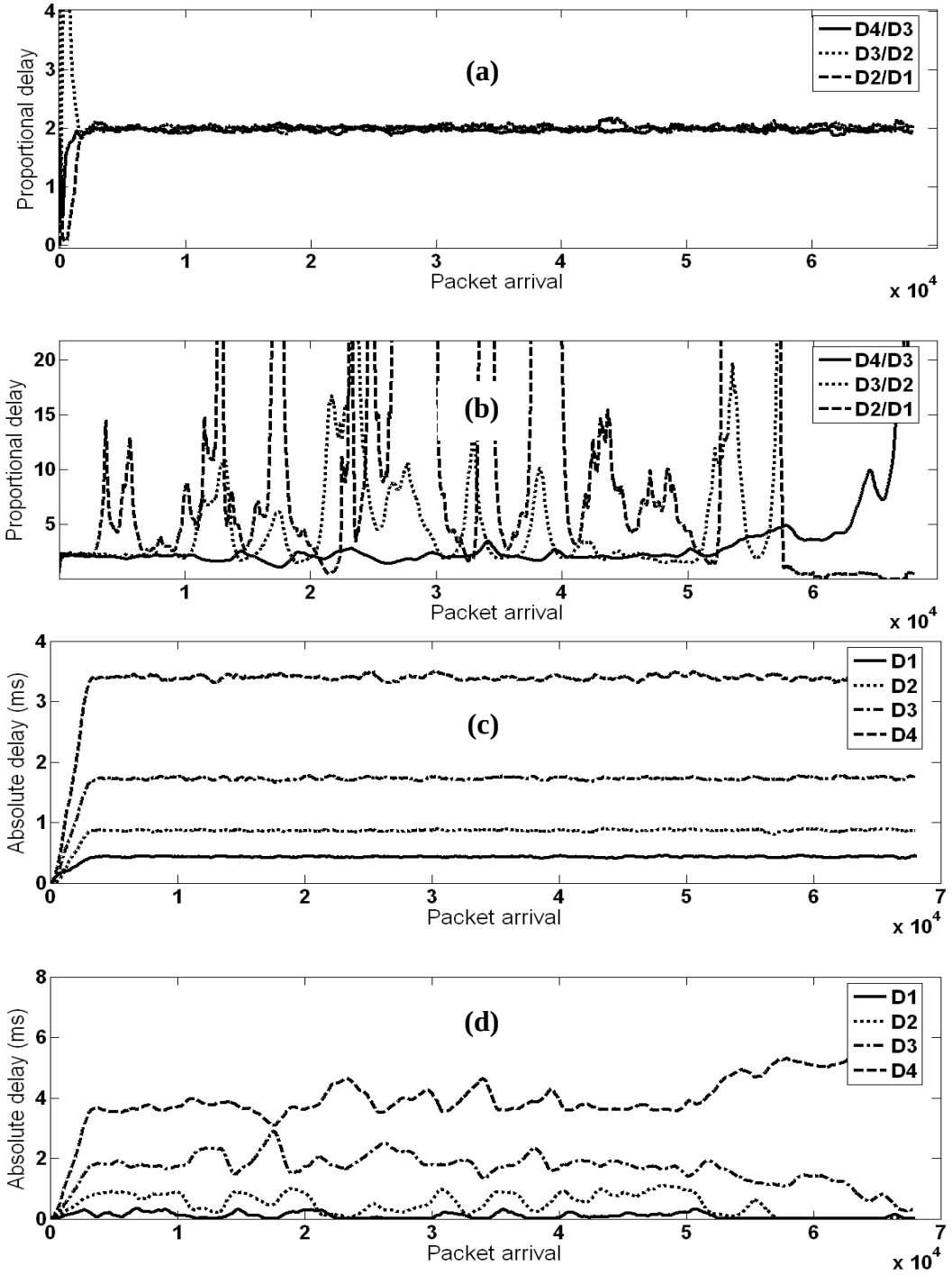

Figure 11. Comparison of AMPCS and JoBS algorithms. The two upper graphs are proportional delays when using (a) AMPCS and (b) JoBS. The two lower graphs show delay values for (c) AMPCS and (d) JoBS. Horizontal axis shows the number of events (Packet arrivals)

The computational cost of this optimization approach varies linearly with the number of optimization variables (here Q). This significantly improves some standard quadratic programming solvers for which the computational cost is cubic in the number of optimization variables. With a complexity order of $O(Q)$, an AMPCS is comparable in complexity to previous algorithms. This algorithm is not used in this paper; however, it is shown in [38] that AMPCS can be implemented in very high speed links as well. However, we measured the simulation run time for two algorithms. The JoBS is about 2.5 times faster than AMPCS, e.g. in one of the simulations, JoBS run time was 72 S, which was faster than 176 S for AMPCS.

Our metrics for comparing two algorithms are:

- Averaged instantaneous proportional delay error. This error is calculated as:

$$E_{ins,i} = \sum_{n=1,D_{i,n}>0,B_{i,n}>1,B_{i+1,n}>1}^{N} \left| (D_{i+1,n} / D_{i,n}) - k_i \right| / N_{busy} / k_i \tag{26}$$

where $D_{i,n}$ is the instantaneous delay of the i'th queue at the n'th arrival, N is total number of packet arrivals, $B_{i,n}$ is the backlog of the *i'th* queue at *n'th* arrival and k_i is the desired value for the proportional delay of i'th and *(i+1)*'th queues. By the instantaneous delay, we mean the delay for the packet, which is currently being serviced. It is important to note that a divide by zero occurs when $D_{i,n}$ is zero. To avoid the divide by zero error and also for the cases that backlog of any of two queues is not more than 1 packet, the sample is omitted from the error calculations. So, the N_{busy} is the number of events that are not omitted from calculations.

- Maximum delay ($D_{Max,i}$). The maximum delay that a packet tolerates during the simulation time. The same traffic load should be applied to both algorithms for a fair comparison.
- Maximum proportional delay error: This metric is the maximum amount for the instantaneous proportional delay error.

$$E_{Max,i} = \underset{n,D_{i,n}>0,B_{i,n}>1,B_{i+1,n}>1}{Max} \left| (D_{+1i,n} / D_{i,n}) - k_i \right| / k_i \tag{27}$$

- Average proportional delay error: This error is calculated as:

$$\overline{E_i} = \sum_{n=1,D_{i,n}>0,B_{i,n}>1,B_{i+1,n}>1}^{N} \left| (\overline{D_{i+1,n}} / \overline{D_{i,n}}) - k_i \right| / N_{Busy} / k_i \tag{28}$$

where $\overline{D_{i,n}}$ is the averaged delay of the i'th queue at the n'th arrival over a window size of 1000, N is total number of packet arrivals and k_i is the desired value for the proportional delay of i'th and *(i+1)*'th queues. N_{Busy}, which is a parameter for avoiding divide by zero error, was described in the previous paragraphs.

Table 3 shows the quantity of metrics for two algorithms and two types of input traffic. Each metric has three or four columns. The three columns under proportional metrics ($E_{inst,i}$, $\overline{E_i}$, $E_{Max,i}$) are related to *i=1, 2, 3*, respectively. The four columns under maximum delay ($D_{Max,i}$) are related to *i=1, 2, 3, 4*, respectively. It can be seen that for the CBR traffic, our algorithm AMPCS gives much better results than JoBS, but for the Pareto traffic, the metrics becomes approximately similar for two algorithms. This variant behavior of the AMPCS algorithm relates to the dynamics of model parameters. In fact, the MPC accuracy is directly proportional to model accuracy. As we employ an adaptive model, the model parameters do not need to change rapidly when the traffic is CBR and the model becomes more accurate than the case when the input traffic is Pareto. However; when the traffic is Pareto, the model parameters change rapidly, so the model does not track the system dynamics as fast.

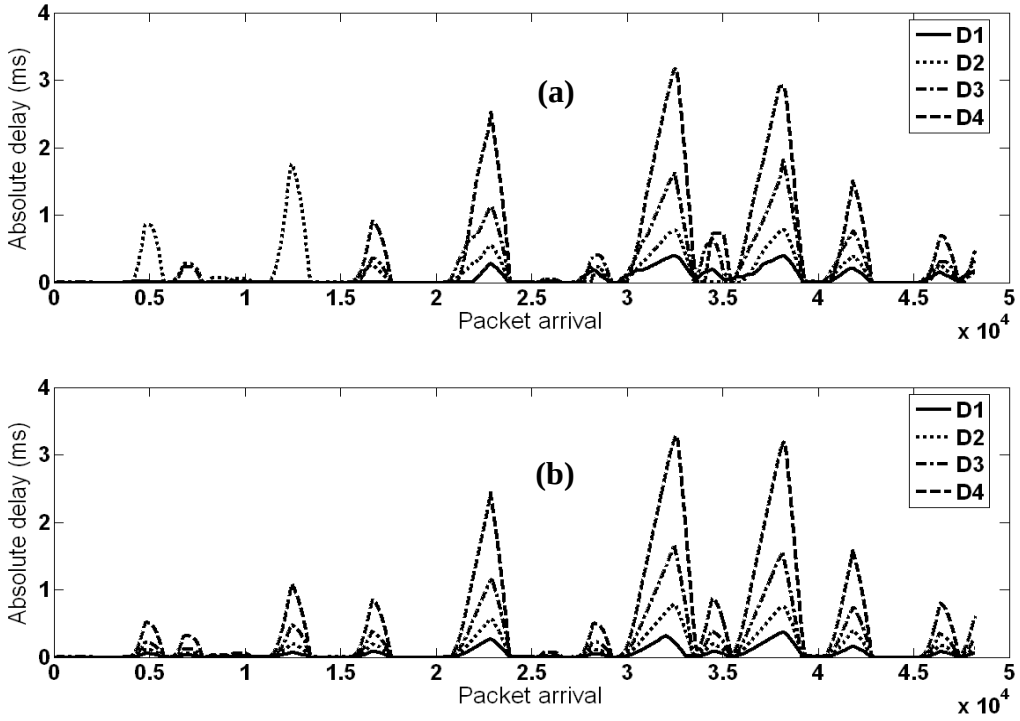

Figure 12. Delay values for Pareto input traffic and applying AMPCS (a) and JoBS (b) algorithms. Horizontal axis shows the number of events (Packet arrivals)

At low connection loads on an uncongested link with little or no queuing, the traffic variables are long-range dependent and bursty. As the load increases, the laws of superposition of marked point processes push the arrivals toward Poisson [8] and the burstiness is reduced. So, our algorithm can act well at the core routers where hundreds of flows are multiplexed. Although it was shown that the performance of class based queues is reduced for the self similar traffic [3], we should consider the adaptation speed of the model in our future works to reduce the error rates regarding bursty traffic.

The simulation results are shown in Figures 11 and 12. The input traffic for the Figure 11 is CBR. The Figures 11.a and 11.b show the proportional delay, while 11.c and 11.d present the delay values. It can be seen that our algorithm controls the proportional delay more accurate than JoBS for the CBR traffic. The delay values for the Pareto input traffic are presented in Figure 12.

The main reason for the difference between the results of our simulation and what Nicols Christin *et. al* present for JoBS in [5] is that they average the proportional delay over 0.5 second periods of time. Averaging over a long period of time for a 10 Gbps link reduces the proportional delay error rate, however, the JoBS gives much better results than the other two algorithms i.e. Waiting Time Priority (WTP) [13] and Mean Delay Proportional (MDP) [32], [15]. We also average the delay for JoBS and AMPCS, but we use a 150 ms time interval.

8. Conclusions and Future Work

A new algorithm was proposed for scheduling problem. The AMPCS algorithm applies a model predictive controller to the scheduling problem. RLS algorithm was utilized to estimate the parameters of the adaptive model. The MPC predicts the future delay of the different service classes based on adaptive model. Solving an optimization problem, MPC finds the optimum service rates for traffic classes. The simulation results show that our new algorithm guarantees absolute and proportional delay and jitter for DiffServ classes. Adaptive modeling can overcome the nonlinearity of the system; therefore, the algorithm is accurate enough for QoS control.

Our search results shows that this is the first time that MPC is applied to scheduling problem in routers. Stability conditions of the closed loop control system should be investigated precisely in the future. Also, MPC implementation problems in high speed links should be considered in our future works.

Acknowledgement

The authors sincerely thank the Iran Telecommunication Research Center (ITRC) for their support and the referees for their insightful comments.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *RFC2475*, Dec. 1998.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," *RFC 1633*, Jun. 1994.

[3] R. Braen, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification," *RFC 2205*, Sept. 1997.

[4] E. Rosen, A. Viswanathan, and R.Callon, "Multiprotocol Label Switching Architecture," *RFC 3031*, Jan. 2001.

[5] N. Christin, J. Liebeherr, and T. F. Abdelzaher, *A quantitative assured forwarding service*, Technical Report, University of Virginia, 2001.

[6] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," *RFC 2598*, 1999.

[7] C. Dovrolis, *Proportional differentiated services for the Internet*, PhD thesis, University of Wisconsin-Madison, December 2000.

[8] J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun, "On the nonstationarity of Internet traffic," *Proc, ACM SIGMETRICS Inter. Conf. on Measurement and modeling of computer systems*, pp. 102-112, 2001.

[9] D. Clark, *Adding Service Discrimination to the Internet*, Internet Economics, the MIT Press, 1997.

[10] I. Stoika, and H.Zhang, "LIRA: An Approach for Service Differentiation in the Internet," *Proc, of 8th Inter. Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 115-128, 1998.

[11] Y. Lu, Tarek F. Abdelzaher, and A. Saxena, "Design, Implementation and Evaluation of Differentiated Caching Services," *IEEE Trans. Parallel and Distributed Systems*, Vol. 15, No. 5, pp. 440-452, 2004.

[12] C. Dovrolis, and P. Ramanathan, "Proportional differentiated services, part II: Loss rate dif ferentiation and packet dropping," *Proc, Inter. Workshop on Quality of Service,* pp. 52–61, 2000.

[13] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional differentiated services: Delay differentiation and packet scheduling," *Proc, ACM SIGCOMM*, pp. 109–120, 1999.

[14] A. K. Parekh, and R. G. Gallagher, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Networking*, Vol. 1, No. 3, pp. 344–357, 1993.

[15] S. Athuralia, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," *IEEE Network,* Vol. 15, No. 3, 2001.

[16] U. Bodin, A. Jonsson, and O. Schelen, "On creating proportional loss differentiation: predictability and performance," *Proc, Inter. Workshop on Quality of Service*, pp. 372–386, 2001.

[17] J. Liebeherr, and N. Christin, "JoBS: Joint buffer management and scheduling for differentiated services," *Proc, Inter. Workshop on Quality of Service*, pp. 404–418, 2001.

[18] A. D. Striegel, and G. Manimaran, "Packet Scheduling with Delay and Loss differentiation," *Computer Communications*, Vol. 25, No. 1, pp. 21-31, 2002.

[19] T. Nandagopal, N. Veinkitaraman, R. Sivakumar, and V. Barghavan, "Delay differentiation and adaptation in core stateless networks," *Proc, IEEE Conf. on Computer Communications*, pp. 421-430, 2000.

[20] S. Keshav. "A control-theoretic approach to flow control," *ACM Compter Communication review*, Vol. 21, No. 4, pp. 3-15, 1991.

[21] Y. Gu, H. O. Wang, Y. Hong, and L. G. Bushnell, "A Predictive Congestion Control Algorithm for High Speed Communication Networks," *Proc, American Control Conference*, Vol. 5, pp. 3779-3780, 2001.

[22] B.-J. Chang, S.-Y. Lin, and J.-Y. Jin, "LIAD: Adaptive bandwidth prediction based Logarithmic Increase Adaptive Decrease for TCP congestion control in heterogeneous wireless networks," *Computer Networks*, Vol. 53, No. 14, pp. 2566-2585, 2009.

[23] F. Delli Priscoli, and A. Pietrabissa, "Control-based Connection Admission Control and Downlink Congestion Control procedures for satellite networks," *Journal of the Franklin Institute*, Vol. 346, No. 9, pp. 923-944, November 2009.

[24] N. Bigdeli, and M. Haeri, "Predictive functional control for active queue management in congested TCP/IP networks," *ISA Transactions*, Vol. 48, No. 1, pp. 107-121, 2009.

[25] F. Blanchini, R. L. Cigno, and R. Tempo, "Robust rate control for integrated services packet networks," *IEEE/ACM trans. networking*, Vol. 10, No. 5, pp. 644-652, 2002.

[26] M. Mahramian, and H. Taheri, "A model predictive control approach for guaranteed proportional delay in DiffServ architecture," *Proc, 10th Asia-Pacific conf. on communications*, pp. 592-597, 2004.

[27] D. W. Clarke, C. Mohtadi and P. S. Tuffs, "Generalized predictive control, Part 1. The basic algorithm," *Automatica*, Vol. 23, No. 2, pp. 137-148, 1987.

[28] S. J. Qin, "An overview of industrial model predictive control technology," *Chemical Process Control*, Vol. 93, No. 316, pp. 232-256, 1997.

[29] E. F. Camacho, and C. Bordons, *Model predictive control*, Springer-Verlag, 1999.

[30] J. M. Maciejowski, *Predictive control with constraints*, Pearson education, 1st ed., 2002.

[31] H. Elliott, and W. A. Wolovich, "A Parameter Adaptive Control Structure for Linear Multivariable Systems," *IEEE Trans. Automatic Control*, Vol. AC-27, No. 2, pp. 340-352, 1982.

[32] S. Mc Canne, and S. Floyd, "UCB/LBNL Network Simulator – ns (version 2)," http://www-mash.cs.berkley.edu/ns/, 1998.

[33] A. Broumandan, N. Rezaee, and M. Mahramian, "Queuing model selection and estimation of differentiated services architecture," *Proc, EUROCON*, pp. 1220-1223, 2005.

[34] L. G. Bleris, J. Garcia, M. V. Kothare, and M. G. Arnold, "Towards Embedded Model Predictive Control for System-on-a-Chip Application," *Journal of Process Control*, Vol. 16, No. 3, pp. 255-264, 2004.

[35] P. Tondel, and A. Johansen, "Complexity Reduction Explicit Linear Model Prediction control," *IFAC World Congress*, 2002.

[36] A. A. Tyagunov, *High Performance Model Predictive Control for Process Industry*, PhD thesis, Eindhoven University of Technology, 2004.

[37] M. Fu, Z. Q. Luo, and Y. Ye, "Approximation algorithms for quadratic programming," *Journal of combinatorial optimization*, Vol. 2, No. 1, pp. 29-50, 1998.

[38] M. Mahramian, H. Taheri, and M. Haeri, "SMPCS: Sub-optimal model predictive control scheduler," *Proc, 6th Inter. Conf. on next generation teletraffic and wired/wireless advanced networking*, Vol. 4003/2006, pp. 554-565, 2006.

[39] J. Orellana, L. Sacks, and H. de Meer, "Performance of class based queues with self similar traffic", *Proc, London Communication Symposium,* 2002.

Mehran Mahramian received the B.Sc. and M.Sc. degrees in Electrical Engineering from Sharif University of Technology, Tehran, Iran in 1995 and 1997 respectively. From 1997, he stayed in Informatics Services Corp. to implement broadband satellite communication equipments. He got PhD degree in Electrical Engineering from Amirkabir University of Technology in 2006.
E-mail: m_mahramian@isc.iranet.net

Hassan Taheri received the B.Sc. degrees in Electrical Engineering from Amirkabir University of Technology. He got his M.Sc. and PhD. in Electrical Engineering from University of Manchester Institute of Science and Technology in 1978 and 1988 respectively. He is now the head of Electronics group at Electrical Engineering Department of Amirkabir University of Technology.
E-mail: htaheri@aut.ac.ir

Masoud Shafiee was born in Ardestan, Iran on August 1, 1957. He received the B.Sc. degree in mathematics in 1978, the M.Sc. degrees in mathematics (1981) and system engineering (1983) from Write State University, Dayton, Ohio, USA, the M.Sc. and Ph.D. degrees in electrical engineering from the Louisiana State University, Louisiana, USA, in 1985 and 1987, respectively. In 1987, he joined the department of electrical engineering at Amirkabir University of Technology. He is currently a full professor of electrical engineering.

His current research interests are in stability of multidimensional systems, singular system and robotics. He has published more than 120 technical papers in these areas. Professor Shafiee is the author of Fourier analysis and its applications in engineering, engineering mathematics, linear algebra, stepper motors, and signal and systems, all in Persian.

E-mail: mshafiee@aut.ac.ir

Paper Handling Data:

Submitted: 01.09.2005
Received in revised form: 14.04.2010
Accepted: 01.04.2006
Corresponding author: Dr. Mehran Mahramian, Electrical Engineering Department, Amirkabir University of Technology, Tehran, Iran.