An Optimal RIO with Statistical Delay Assurances

Rajesh Kumar Patro and Shalabh Bhatnagar
Department of Computer Science and Automation
Indian Institute of Science
Bangalore-560012, India
{prajesh,shalabh}@csa.iisc.ernet.in

***Abstract*— DiffServ uses the RIO active queue management technique to provide diferentiated service to the different aggregated traffic classes. The parameter setting of RIO dictates the behaviour of the router and thus the overall QoS performance of the network router. We develop an optimization framework that strives to guarantee statistical delays in a RIO router. We solve the optimization problem using nonlinear programming techniques, viz., penalty and barrier methods. The gradient and Hessian terms required in the optimization procedure are obtained using a novel adaptive four-timescale stochastic approximation algorithm and a modified second-order SPSA parameter update is then performed. We demonstrate the effectiveness of the framework by discussing the results of the simulations that we have performed. We observe that the proposed framework is an effective mechanism to ensure statistical delay guarantees while functioning with an optimal set of parameters in RIO routers.**

I. Introduction

The present day Internet provides best-effort service without any service guarantees. However, even the best behavior on the part of the network may still fall short of the requirements of today's applications. Modern day real-time applications such as audio-on-demand, video-on-demand, real-time streaming etc. impose stringent quality of service (QoS) demands on the network.

The IP-packets queued up at the routers are managed using adaptive queue management (AQM) techniques. An AQM technique that has attracted a lot of research over the last few years is the random early detection (RED) scheme [4]. Traditionally, RED has been used to control network congestion. However recently, proposals have been suggested to demonstrate RED's utility in the QoS domain. One such proposal is the RED *In/Out* (RIO) [3] algorithm. Essentially, it uses two sets of RED parameters for two different types of traffic. RIO can also be suitably extended to service the various aggregated traffic classes of the differentiated services (DiffServ) architecture. DiffServ has been the preferred approach towards attaining QoS in the Internet as it only adds minimal complexity to the existing routers and thus offers a smooth transition from a *best-effort* network to a *QoS differentiated* network. DiffServ aggregates similar traffic into predefined traffic groups by appropriately classifying packets at the *edge-routers* i.e., the first-hop routers. A packet belonging to a particular traffic aggregate is identified and marked with a differentiated services code point (DSCP). This code defines the differential treatment to be meted out to the packets at the DiffServ routers. This is referred to as the Per Hop Behavior (PHB). One of the standard PHB's is the Assured Forwarding PHB (AF-PHB) in which the connections only receive statistical guarantees on the service. The priority (DSCP code) of a packet determines its relative importance during periods of congestion. The high priority (low drop-precedence) packets are given preference over the low priority (high drop-precedence) ones. Each precedence level is assigned their respective set of RED parameters (w_q, max_p, min_{th}, max_{th}) [4].

At the start of a connection, a QoS agreement is decided between the source and the network defined via a service level agreement (SLA). Packets that conform to this agreement are marked as *In*-profile whereas packets that violate their agreements are marked as *Out*-profile ones. RIO reserves a separate set of RED parameters for each packet profile. The *In* and *Out* packets are considered to be in two virtual queues and managed by their respective sets of RED parameters. A gentle set of RED parameters is used for servicing *In*-profile packets while a harsher set of parameters is employed for the *Out*-profile ones. The packet marking probability parameter, max_p, for *In*-profile is lower than that for *Out*-profile. Similarly, the queue threshold parameters (min_{th} and max_{th}) for *In*-packets are higher than their *Out*-profile counterparts. This ensures that the conformant (high-priority) packets receive better service.

The parameter setting of RIO has a huge influence on its overall performance [8], [3]. A sub-optimal RIO parameter setting has been shown to result in under utilization of resources and a degradation in its performance. In this paper, we develop a mathematical optimization framework that outlines a methodology aimed at the optimization of RIO parameters. The optimization problem is solved using nonlinear programming techniques viz., penalty and barrier methods. An adaptive four-timescale stochastic approximation algorithm has been developed that estimates the various gradient and Hessian quantities involved in the optimization process. A modified second-order SPSA parameter [9] update is used

to avoid the complexities involved in computing the inverse of the Hessian matrix as well as overcome any slowness in convergence that can happen if the Hessian is ill-conditioned. In the proposed framework, we aim to stabilize the queue variations around a pre-defined target, Q^*, that indirectly guarantees a statistically assured delay while also ensuring that the network resources do not go under-utilized. As we shall see, this method of optimal parameter tuning enhances the performance of RIO.

The rest of the paper is organized as follows : Section II explains the proposed delay assurance model, the associated nonlinear optimization procedures and the assumptions therein. In Section III, we describe our adaptive four-timescale stochastic approximation algorithm. In Section IV, we present the results of our simulations. Section V presents the concluding remarks.

II. Optimization Framework and Nonlinear Programming

A. Statistical Delay Assuring Framework

Delay is integral to any QoS framework. In this section, we discuss a mathematical framework which guarantees statistical delay assurances in DiffServ using RIO. In our aim to guarantee delay bounds we also need to address how the other performance metrics (throughput, loss etc.) are affected. Surely, any delay assuring QoS model cannot compromise with these performance parameters. It is well known that the performance of a router is largely affected by the settings of the queue parameters. In our proposed delay assuring framework, we tune the RIO parameters using nonlinear programming methods.

We first state our assumptions (A)-(C) below that assist us in developing the proposed framework and solving it subsequently. In the following, we refer to the steady-state average queue length as simply the average queue length.

Assumptions :

A) Both the mean $E(q_{av})$ and variance $Var(q_{av})$ of the average queue length are twice differentiable and have bounded third derivatives.

B) The average queue length, q_{av}, of a queue has a nonlinear functional relationship with RIO parameters i.e., $q_{av} = f_1(\theta_1, \ldots, \theta_R)$ where R is the number of sets of RED parameters, $\theta_k = (w_{q_k}, max_{p_k}, min_{th_k}, max_{th_k})$, $k = 1, \ldots, R$ and f_1 is a nonlinear function. We thus consider a general case of RIO parameters as comprising R sets of RED parameters.

C) The average queue length, q_{av} is normally distributed with mean $\hat{q}_1 = E(q_{av})$ and variance $\hat{\sigma}^2 = Var(q_{av})$ i.e. $q_{av} \sim N(\hat{q}_1, \hat{\sigma}^2)$.

We now formulate the delay assuring optimization problem. The objective of the optimization is to minimize the mean squared error of average queue length, q_{av}, from a target queue size of Q^*. The minimization of this objective function is subject to the constraint that the probability of q_{av} being less than or equal to the target queue length Q^* should not drop below α. The optimization problem can be mathematically stated as follows:

$$\begin{aligned} &\min_{\theta} E(q_{av} - Q^*)^2 \\ &\text{s.t } P(q_{av} \leq Q^*) \geq \alpha \end{aligned} \tag{1}$$

where C_l - Link capacity of the RIO router, T_d - Average delay at the RIO router, Q^* - Target steady state average queue length at the RIO router and $\alpha \in [0, 1]$. Also, $Q^* = C_l T_d$.

The optimization problem (1) can be reworked as shown :

$$\begin{aligned} &\min_{\theta} \ (Eq_{av}^2 + {Q^*}^2 - 2Q^* Eq_{av}) \\ &\text{s.t } P\left(\frac{q_{av} - Eq_{av}}{\sqrt{Var(q_{av})}} \leq \frac{C_l T_d - Eq_{av}}{\sqrt{Var(q_{av})}}\right) \geq \alpha \end{aligned}$$

Defining $\hat{q}_2 = Eq_{av}^2$ and using Assumption (C), the above problem can be further reduced to :

$$\begin{aligned} &\min_{\theta} \ (\hat{q}_2 + {Q^*}^2 - 2Q^* \hat{q}_1) \\ &\text{s.t} \quad C_l T_d - \hat{q}_1 \geq \phi^{-1}(\alpha) \ \hat{\sigma} \end{aligned}$$

where $\phi^{-1}(\alpha)$ is the inverse of the c.d.f. of the normal distribution evaluated at α. Finally the optimization problem can be written in the following form :

$$\begin{aligned} &\min_{\theta} \ \hat{q}_2 + {Q^*}^2 - 2Q^* \hat{q}_1 \\ &\text{s.t} \ \ C_4 - C_3 \hat{\sigma} - \hat{q}_1 \geq 0 \end{aligned} \tag{2}$$

where $C_4 = C_l T_d$ and $C_3 = \phi^{-1}(\alpha)$.

B. Nonlinear Programming

The format of the optimization problem (2) and assumption (B) allow us to use standard nonlinear programming techniques to solve it. We briefly discuss the barrier and the penalty function methods that we have employed.

1) Penalty Method: We have implemented here the quadratic penalty method [7] to solve the optimization problem (2). The objective function for the unconstrained optimization problem is generated after augmenting the quadratic penalty term as shown :

$$P(\theta; r) = \hat{q}_2 + {Q^*}^2 - 2Q^* \hat{q}_1 - \frac{1}{2r}(C_4 - C_3 \hat{\sigma} - \hat{q}_1)^2 \tag{3}$$

and the problem is now reduced to solving an unconstrained optimization problem i.e., $\min_{\theta} P(\theta; r)$ for a sequence of values of $r = r_k$, $r_k \uparrow \infty$. The gradient and the Hessian of $P(\theta; r)$ is given as follows : For $r = r_k$,

$$\nabla_\theta P = \hat{q}_2^{'} - 2Q^* \hat{q}_1^{'} + \frac{1}{r_k}[(C_4 - C_3 \hat{\sigma} - \hat{q}_1)(C_3 \hat{\sigma}^{'} + \hat{q}_1^{'})] \tag{4}$$

$$\nabla^2_\theta P = \hat{q}_2^{''} - 2Q^*\hat{q}_1^{''} + \frac{1}{r_k}[K - (C_3\hat{\sigma}^{'} - \hat{q}_1^{'})^2] \quad (5)$$

where $K = (C_4 - C_3\hat{\sigma} - \hat{q}_1)(C_3\hat{\sigma}^{''} + \hat{q}_1^{''})$.

2) Barrier Method: The standard logarithmic Barrier method has been employed to solve problem (2). The unconstrained optimization objective function thus obtained is given by :

$$B(\theta; b) = \hat{q}_2 + Q^{*2} - 2Q^*\hat{q}_1 - b\log[C_4 - C_3\hat{\sigma} - \hat{q}_1] \quad (6)$$

and the problem is now reduced to solving an unconstrained optimization problem, $\min_\theta B(\theta; b)$ for a sequence of values of $b = b_k$, $b_k \downarrow 0$. Similarly, the gradient and the Hessian of $B(\theta; b)$ is given as follows : For $b = b_k$,

$$\nabla_\theta B = \hat{q}_2^{'} - 2Q^*\hat{q}_1^{'} + b_k\frac{(C_3\hat{\sigma}^{'} + \hat{q}_1^{'})}{(C_4 - C_3\hat{\sigma} - \hat{q}_1)} \quad (7)$$

$$\nabla^2_\theta B = \hat{q}_2^{''} - 2Q^*\hat{q}_1^{''} + b_k\frac{[K + (C_3\hat{\sigma}^{'} - \hat{q}_1^{'})^2]}{(C_4 - C_3\hat{\sigma} - \hat{q}_1)^2} \quad (8)$$

III. Four Timescale Stochastic Approximation

The non-availability of the functional relationship between RIO (RED) parameters and the average queue length, q_{av}, makes it impossible to directly estimate the required gradient and Hessian estimates of the penalty and the barrier objective functions. Traditionally, simulation based algorithms have been used to solve such problems where the analytic ralationship between the objective function and the system input parameters is not known. We have devised an adaptive four-timescale stochastic approximation algorithm for obtaining various gradient and Hessian estimates and use these for performing parameter updates. Four timescales with stepsizes $d(n)$, $b(n)$, $c(n)$ and $a(n)$, respectively have been employed. These stepsizes satisfy stadard properties, see [1]. The quantities $\hat{q}_1$, $\hat{q}_2$ and $\hat{\sigma}$ are averaged over the fastest timescale, $d(n)$. This is required since we want the above quantities to have averaged out before gradient and Hessian are computed. Subsequently, the gradient and Hessian updates are performed using the next two timescales, $b(n)$ and $c(n)$, respectively. The Hessian updates are performed on the faster timescale than gradient updates since one requires these to have converged and the inverse of the Hessian taken before each gradient update step. Finally, the parameters are updated over the slowest timescale, $a(n)$, using a modified second-order SPSA based approach. We now discuss the details of the algorithm to optimize RIO parameters used in DiffServ. Here, we address the same for the penalty approach only, the barrier method implementation being completely analogous.

For R sets of RED parameters (i.e. one set ot RIO parameters), first fix $\theta_k(0) = (\theta_{k1}(0), \ldots, \theta_{k4}(0))^T$ for $k = 1, \ldots, R$. An 8×4 perturbation sequence matrix, $\hat{H}$, is then formed from an 8×8 normalized Hadamard matrix (see [2]), H by choosing any four of its columns. Now set $\Delta_k(n) = \hat{H}(n \bmod 8 + 1)$ and $\hat{\Delta}_k(n) = \hat{H}(8 - n \bmod 8)$ and perform four simulations using perturbations $\theta_k^+(n) = (\theta_{k_i}(n) + \delta_1\Delta_{k_i}(n),\ i = 1, \ldots, 4)$, $\theta_k^-(n) = (\theta_{k_i}(n) - \delta_1\Delta_{k_i}(n),\ i = 1, \ldots, 4)$, $\theta_k^{-+}(n) = (\theta_{k_i}(n) - \delta_1\Delta_{k_i}(n) + \delta_2\hat{\Delta}_{k_i}(n),\ i = 1, \ldots, 4)$, and $\theta_k^{++}(n) = (\theta_{k_i}(n) + \delta_1\Delta_{k_i}(n) + \delta_2\hat{\Delta}_{k_i}(n),\ i = 1, \ldots, 4)$, respectively for $k = 1, \ldots, R$ and (small) constants δ_1, $\delta_2 > 0$. Using the fastest timescale $d(n)$, the first and the second-order moments of q_{av} are averaged on every packet arrival. The priority of each packet is identified from the packet header and the corresponding parameter set k ($k \in \{1, \ldots, R\}$) is selected. The averaging is performed using,

$$Z^w_{q_v}(nL+m+1) = Z^w_{q_v}(nL+m) + d(n)\left(\mu^v - Z^w_{q_v}(nL+m)\right),$$

where $\mu^v = (q_{av}(nL+m))^v$, $v \in \{1, 2\}$ and $w \in \{-, +, -+, ++\}$. The variance is then estimated using, $\hat{\sigma}^w(n_kL + m) = \sqrt{Z^w_{q_2}(n_kL+m) - (Z^w_{q_1}(n_kL+m))^2}$.

The required gradient estimates are averaged over the next fastest timescale, $b(n)$. For $(W, X, Y) \in \{(Z_{q_1}, \hat{q}_1^{'}, \hat{q}_1^{''}), \{(Z_{q_2}, \hat{q}_2^{'}, \hat{q}_2^{''}),\ (\hat{\sigma}, \hat{\sigma}^{'}, \hat{\sigma}^{''})\}$, $w \in \{-, +, -+, ++\}$ and all $i \in \{1, \ldots, 4\}$,

$$Y_i(nL + m + 1) = Y_i(nL + m) + b(n)(V_1 - Y_i(nL + m))$$

where $V_1 = \frac{W_i^+(nL+m) - W_i^-(nL+m)}{2\delta_1\Delta(n)}$. Now the Hessian estimates are obtained using the next timescale, $c(n)$. For each $j, i \in \{1, 2, 3, 4\}$, update

$$X_{j,i}(n+1) = X_{j,i}(n) + c(n)\left(V_2 - X_{j,i}(n)\right)$$

where $V_2 = \frac{\left(\frac{W_i^{++}(nL) - W_i^+(nL)}{\delta_2\hat{\Delta}(n)}\right) - \left(\frac{W_i^{-+}(nL) - W_i^-(nL)}{\delta_2\hat{\Delta}(n)}\right)}{2\delta_1\Delta(n)}$. The terms $\hat{q_1}$, $\hat{q_2}$ and $\hat{\sigma}$ are obtained using, $V_3(nL) = \frac{1}{2}(V_3^+(nL) + V_3^-(nL))$, for all $V_3 \in \{\hat{q_1}, \hat{q_2}, \hat{\sigma}\}$.

The various estimates thus obtained are then substituted in (4) and (5) to estimate the gradient, $\nabla_\theta P$, and Hessian, $\nabla^2_\theta P$, of the penalty objective function. In place of the inverse of the Hessian, we have employed a modified second-order parameter update. This aids in suppressing the effects of any ill-conditioning in the Hessian that can slow down the optimization process as well as eliminating the complexities involved in computing the inverse of the Hessian matrix. Using the eigen-values (λ_i, $i = 1, \ldots, 4$ such that $\lambda_i > \lambda_{i+1}$) of $\nabla^2_\theta P$, a diagonal matrix, $\Pi_n \equiv \text{diag}\,[\lambda_{1,n}, \cdots, \lambda_{q,n}, \lambda_{q+1,n}, \cdots, \lambda_{4,n}]$ is formed. Further, $\lambda_{q,n} > 0$, and $\lambda_{q+1,n} \leq 0$, for $q \in \{1, \cdots, 4\}$. The negative eigen-values along with the smallest positive eigen-value is then mapped to the positive set according to $\hat{\lambda}_{q,n} = \eta\lambda_{q-1,n}, \hat{\lambda}_{q+1,n} = \eta\hat{\lambda}_{q,n}, \cdots, \hat{\lambda}_{4,n} = \eta\hat{\lambda}_{3,n}$, where $\eta = (\frac{\lambda_{q-1,n}}{\lambda_{1,n}})^{q-2}$. If all $\lambda_{j,n} > 0, j \in 1, \ldots, 4$, set $\hat{\lambda}_{j,n} = \lambda_{j,n}$. The geometric mean, $\hat{\lambda}_n$, of the mapped eigen-values is evaluated using $\hat{\lambda}_n = \left[\lambda_{1,n}\lambda_{2,n}\cdots\lambda_{q-1,n}\hat{\lambda}_{q,n}\hat{\lambda}_{q+1,n}\cdots\hat{\lambda}_{4,n}\right]^{\frac{1}{4}}$ and

finally, for $i = 1, \ldots, 4,$, the parameter is updated over the slowest timescale, $a(n)$, using the recursion,

$$\theta_{k_i}(n+1) = \Gamma_{k_i}\left(\theta_{k_i}(n) - a(n)(\hat{\lambda}_n)^{-1}\hat{\nabla}_{\theta_{k_i}(n)}P\right),$$

where $\Gamma_{k_i} : \mathcal{R} \rightarrow [a_{k_i}, b_{k_i}]$ defined by $\Gamma_{k_i}(x) = \max(\min(x, b_{k_i}), a_{k_i})$ is the projection operator that projects any $x \in \mathcal{R}$ to the constraint interval $[a_{k_i}, b_{k_i}]$, $a_{k_i} < b_{k_i}$ corresponding to parameter θ_{k_i}.

The penalty parameter, r_k is updated using $r_{k+1} = 10r_k$. Similarly, the barrier parameter, b_k, is updated using $b_{k+1} = 0.1b_k$ (see [7] for a detailed discussion on the stopping criterion and convergence of penalty algorithms). Our algorithm gives appropriate convergence behavior because of the difference in timescales as discussed earlier. The general idea used is that a faster timescale recursion sees the slower recursion as quasi-static while the slow one sees the faster recursion as essentially equilibrated.

IV. Simulation Results

We now present the results of the simulation experiments that we have carried out. All the simulations were performed using the *ns2.26* simulator [5]. The network topology is as shown in Figure 1. RIO is configured over the bottleneck link between $C0$ and $E1$. Multiple (10) infinite-FTP connections are connected to each of the source nodes $S0$, $S1$, $S2$. Node $S3$ has a non-cooperating UDP-connection established with $D0$. The simulations were conducted for multiple SLA commitments ranging from 25% to 125% of the bottleneck bandwidth reserved by all the sources. We have considered here experiments where a fixed load for the entire duration of the simulation is used and also those for which load dynamically increases (adding a new source node with multiple connections) every 200 seconds. The initial parameter settings have been chosen as $min_{th} = 20, max_{th} = 60, w_q = 0.001, max_p = 0.01$ for the *In*-profile packets and $min_{th} = 10, max_{th} = 30, w_q = 0.005, max_p = 0.1$ for the *Out*-profile packets, respectively.

We compare the various performance metrics (queue variation, delay, jitter, throughput, loss rate) of RIO with those of the proposed barrier-optimized RIO (B-RIO) and penalty-optimized RIO (P-RIO). Figures 2(a), 2(b) and 2(c), respectively, show the queue variations for RIO, B-RIO and P-RIO for fixed loads. The target queue size in this case is fixed at $Q^* = 40$ for B-RIO and P-RIO. The average buffer occupancy is low in RIO which may not always be desirable as this may lead to under utilization of the available resources, thus highlighting the fact that the parameters may not be optimal. However, the queue variations for B-RIO and P-RIO (see Figures 2(b) and 2(c)) are much less as compared to those of RIO. This is also visible in the changing load scenario for B-RIO and P-RIO (shown in Figures 3(b) and 3(c) respectively). It may be noted that the sizes of the step shifts observed in RIO (see Figure 3(a)) are much less in the case of B-RIO and P-RIO at instants of load increase and the average queue length hovers in a narrow region around the target Q^*. Thus both B-RIO and P-RIO algorithms adapt to the instantaneous load increase by continuing to stabilize in an efficient manner, the queue length around Q^*. The delay plot of Figure 4(a) also highlights this argument. It may be observed here that the *desired* delay in B-RIO and P-RIO is stable even with changing network loads whereas the delay bounds for RIO increase with increasing bandwidth demands. Moreover, the fluctuations in the instantaneous queue lengths are also large in RIO leading to higher jitter variations. This can be seen from Figure 4(b). The optimal auto-tuning of the parameters using P-RIO and B-RIO also results in an improved throughput (Figure 5(a)) and packet drop rates (Figure 5(b)). The drop rate is reduced in P-RIO and B-RIO owing to better (optimal) buffer usage (see Figure 5(b)). This also manifests in an improved throughput performance for the optimized RIO algorithm. The throughput for B-RIO and P-RIO can be seen to be consistently higher than that of RIO, see Figure 5(a). This is attributed to the optimal auto-tuning of the RIO-parameters performed by the above procedures.

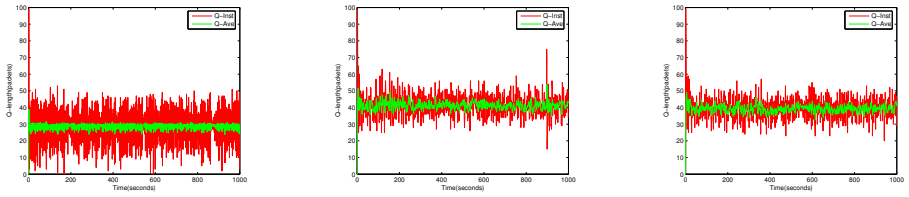
V. Conclusions

We have shown in this paper that the proposed nonlinear optimization framework is able to guarantee stabilized delays in RIO-routers. We have used the penalty and the barrier methods to solve the optimization problem. An adaptive four-timescale stochastic approximation algorithm has been developed to perform the optimization. The proposed algorithm is able to auto-tune the RIO-parameters according to the current network conditions as our numerical results under various network/traffic conditions demonstrate. The effectiveness of our algorithm can be verified through the improved performance of RIO using the proposed optimization framework. We also demonstrated that delay bounds in DiffServ using RIO can be assured within the optimization framework.

We now state a few interesting extensions to this work. The proposed optimization framework (with the underlying nonlinear optimization) is generic in nature, in that, it can be easily extended to any QoS framework. Nonlinear techniques other than the methods that we have implemented, may also be tried out and a comparative work on the different approaches to nonlinear programming would be worth investigating. Other stochastic approximation techniques can also be implemented to compare performance with our algorithm.

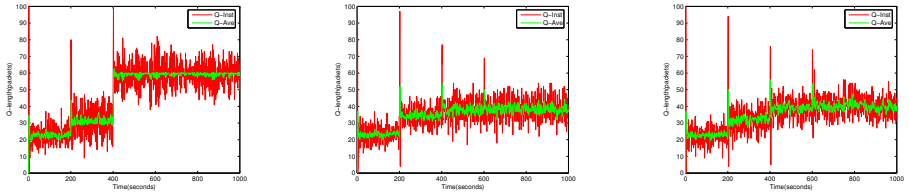
References

[1] Bhatnagar, S. "Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization". ACM Transactions on Modeling and Computer Simulation, Vol 3, No.1 : pp. 74-107, 2005.



(a) RIO (b) B-RIO (c) P-RIO

Fig. 2. RIO Queue Variations (Fixed Load)



(a) RIO (b) B-RIO (c) P-RIO

Fig. 3. RIO Queue Variations (Increasing Load)

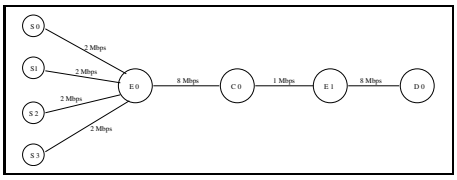
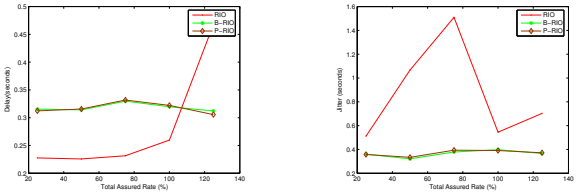
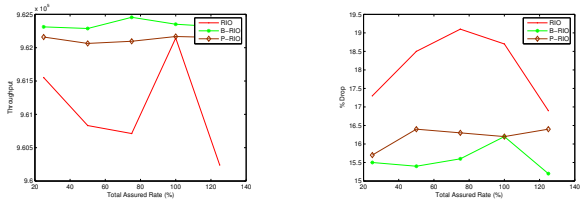


Fig. 1. Topology



(a) Delay (b) Jitter

Fig. 4. Delay and Jitter



(a) Throughput (b) Drop

Fig. 5. Throughput and Drop

[2] Bhatnagar, S., Fu, M. C., Marcus, S. I. & Wang, I. J. "Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences". ACM Transactions on Modeling and Computer Simulation, Vol. 13, No. 4 : pp. 180-209, 2003.

[3] Clark, D. D., & Fang, W. "Explicit allocation of best-effort packet delivery service". IEEE/ACM Transactions on Networking, Vol. 6, No. 4 : pp. 362-373, 1998.

[4] Floyd, S., & Jacobson, V. "Random early detection gateways for congestion avoidance". IEEE/ACM Transactions on Networking, Vol. 1, No. 4 : pp. 397-413, 1993.

[5] ns2 simulator (2002). Network simulator. http://www-mash.cs.berkley.edu/ns, UCB/LBL/VINT.

[6] Orozco, J., & Ros, D. "An adaptive RIO (A-RIO) queue management algorithm". *In Lecture Notes in Computer Science, Springer*.

[7] Rao, S.S. 1996. *Engineering Optimization*. John Wiley & Sons, Inc.

[8] Vaidya, R., & Bhatnagar, S. "Optimized RIO for DiffServ networks". Proceedings of International Conference on Information and Computer Science (ICICS), Dhahran, Saudi Arabia, 2004.

[9] Zhu, X., & Spall, J. C. "A modified second-order SPSA optimization algorithm for finite samples". International Journal of Adaptive Control and Signal Processing, Vol. 16 : pp. 397-409, 2002.