Adaptive Provisioning of Differentiated Services Networks based on Reinforcement Learning

T C K Hui and C K Tham
Dept of Electrical & Computer Engineering
National University of Singapore

to appear in *IEEE Transactions on Systems, Man & Cybernetics* - Part C Applications, Special issue on Technologies that promote computational intelligence, openness and programmability in networks and Internet services, ed.s W. Pedrycz, A. Vasilakos & S. Karnouskos, IEEE, Autumn 2003.

Adaptive Provisioning of Differentiated Services Networks based on Reinforcement Learning

Timothy Chee-Kin Hui, Chen-Khong Tham
National University of Singapore
{engp1698,eletck}@nus.edu.sg

***Abstract*—The issue of bandwidth provisioning for Per Hop Behavior (PHB) aggregates in Differentiated Services (DiffServ) networks has received a lot of attention from researchers. However, most proposed methods need to determine the amount of bandwidth to provision at the time of connection admission. This assumes that traffic in admitted flows always conforms to pre-defined specifications, which would need some form of traffic shaping or admission control before reaching the ingress of the domain. This paper proposes an adaptive provisioning mechanism based on reinforcement-learning principles, which determines at regular intervals the amount of bandwidth to provision to each PHB aggregate. The mechanism adjusts to maximize the amount of revenue earned from a usage-based pricing model. The novel use of a continuous-space, gradient-based learning algorithm, enables the mechanism to require neither accurate traffic specifications nor rigid admission control. Using *ns-2* simulations, we demonstrate using Weighted Fair Queuing, how our mechanism can be implemented in a DiffServ network.**

***Index terms*— Reinforcement Learning, Adaptive bandwidth provisioning, Active networks, Quality of Service, Differentiated Services.**

I. Introduction

Differentiated Services (DiffServ) [1,2] has been widely accepted as the service model to adopt for providing Quality of Service (QoS) over next-generation IP networks. The framework introduces the concept of Per Hop Behaviors (PHBs), which provides different levels of QoS to aggregated flows. This is done by classifying individual traffic flows into various service levels desired before entering the DiffServ network domain. Within the DiffServ domain, flows of the same class are aggregated and treated as one flow. Each aggregated flow is given a different treatment, in terms of network resources assigned, as described by the PHB for that class. There are two PHB groups described for DiffServ; Expedited Forwarding (EF) [3] and Assured Forwarding (AF) [4]. Service level agreements (SLAs) [5] have been proposed to be used in the DiffServ framework to provide network providers a basis to adjust network parameters to meet service level specifications (SLSs) contracted with the users. SLSs contain delay and throughput requirements among others like reliability requirements. Delay requirements are specified in terms of an upper bound, where each packet is to have end-to-end delay less than the value specified. Throughput requirements are specified in terms of a lower bound, where the rate of traffic delivered measured at regular intervals is to be greater than the value specified.

Premium service [6] and Assured service [7] are two levels of service defined in the DiffServ framework to aid in the drawing out of SLAs. Most user applications that require QoS fall into one of these two broad categories. Apart from mission-critical applications such as real-time control where tight delay bounds are needed, a large proportion of real-time applications are non-critical; where users specify certain tolerance levels but may not mind the occasional breech. Examples of such applications are Voice over IP (VoIP), online transactions and video conferencing. This provides room for more flexible QoS mechanisms to be implemented. Bandwidth provisioning is a QoS mechanism that can benefit from this.

Bandwidth provisioning in DiffServ networks involves the determination of the amount of bandwidth to allocate for each PHB aggregate across each network link. This is usually done at the router's outgoing ports through packet scheduling. By provisioning bandwidth, each PHB aggregate shares the bandwidth in a certain proportion as they contend for the use of a network link to transmit data packets from one node to another. By allocating different proportions of bandwidth, the service levels of each PHB can be differentiated. Unfortunately, the proportion of bandwidth to provision is a complex decision due to the interaction of a variety of factors, such as the traffic mix, the level of QoS required and other QoS mechanisms inter-working in the network. For this reason, bandwidth provisioning needs to be adaptive.

Most current implementations adjust provisioning each time new requests are made. This has problems of scalability. Through complex admission control mechanisms, the requests may be accepted or rejected. For the accepted flows, the bandwidth to provision is computed. To accomplish these, the traffic is usually assumed to follow a specified characteristic that is known *a priori*. This is inflexible and often not possible as usage is unpredictable.

Most of the time, after admission control, the provisioning is left static regardless of the traffic dynamics. Some methods proposed are adaptive to the extent of alleviating congestion in the network when detected. Congestion control however does not directly take into consideration feedback of the extent to which QoS is being breeched. The effectiveness of a bandwidth provisioning scheme, just like any other QoS mechanism, is determined by its ability to ensure QoS levels contracted in SLAs are not breeched. At the same time, a network operator desires to maximize its revenue by maximizing the utilization of the network. These two objectives may run contrary to each other, and thus a trade-off has to be made. *Intelligent* bandwidth provisioning should be able to adapt and strike a fine balance, keeping both the users and network providers satisfied.

In this paper, we propose the novel use of a continuous-space, gradient-based reinforcement learning method that makes use of a usage-based pricing approach as a means to adjust PHB aggregate bandwidth provisioning to maximize revenue earned. To maximize revenue, our proposed *intelligent* scheme adaptively finds a policy that strikes a balance between increasing utilization and minimizing QoS penalties.

The original contributions of the proposed scheme presented in this paper are

1. the novel use of continuous-space, gradient-based reinforcement learning to control router bandwidth provisioning in DiffServ networks, thereby enabling a proactive policy to be learned that achieves long-term maximization, rather than short-term gain, as commonly seen in other reactive control-theoretic approaches;
2. the construction of a usage-based pricing plan, which has a variable QoS penalty refund component, that could possibly bring down costs for users as well as increase revenue for network providers;
3. the use of the usage-based pricing plan coupled with an SLA as a feedback to the learning agent, allowing for adaptation to both QoS targets as well as fluctuation in pricing;

The organization of this paper is as follows. In the next section, we describe the challenge of bandwidth provisioning in DiffServ networks and survey some existing methods that have been proposed to solve the problem. In Section III, we describe the theory behind a continuous-space, gradient-based reinforcement learning method that is part of a class of REINFORCE algorithms presented by Williams [8]. In Section IV, we describe our scheme called Reinforcement Learning-based Adaptive Provisioning (RLAP), that implements REINFORCE Gaussian units. RLAP dynamically changes the bandwidth provisioning in routers throughout the DiffServ network. From the authors' knowledge, this is the first such attempt at using continuous-space reinforcement learning in network resource management. We then demonstrate the use of RLAP through the setup of *ns-2* [9] simulations in Section V. The results of multiple simulations are presented for discussion in Section VI. Finally, we discuss several additional issues related to the implementation of RLAP in actual networks in Section VII and conclude in Section VIII.

II. Bandwidth Provisioning In DiffServ Networks

A. Background

In the DiffServ framework, the amount of bandwidth to provision for each PHB aggregate is determined by the service level requirements as stated in the SLS. Often the amount of bandwidth to provision is proportional to the strictness of the requirements, i.e., the lower the delay bound and the higher the throughput bound, the more bandwidth needs to be provisioned. The EF PHB is thus given more bandwidth than is needed (over-provisioned) as it has the strictest of requirements. The AF PHB is on the other hand only slightly over-provisioned as it has more elastic requirements. BE traffic usually gets served with the remaining capacity. Although this is a widely-used method, it requires complex analysis that needs to be based on a variety of factors, such as traffic characteristics, QoS requirements and QoS mechanisms in the network, in order to determine the exact level of provisioning.

To provision bandwidth, it is common to use weighted fair bandwidth allocation, in the form of Weighted fair queuing (WFQ) [10], as it has much better bandwidth utilization under varying traffic conditions as compared to strict bandwidth partitioning. The amount of bandwidth provisioned by WFQ across a link for a particular flow i follows the equation

$$B_i = \frac{\psi_i}{\sum_{j=1}^{n} \psi_j} R \qquad (1)$$

where ψ_i is the weight given to flow i, n is the number of flows utilizing the link and R is the link rate. In our paper, we make use of WFQ as a means of provisioning bandwidth by treating each PHB aggregate as an aggregated flow. As our scheme focuses on adaptive bandwidth provisioning, the scheme may also be suitably applied to other variants of WFQ [11,12].

B. Related Works

The many methods proposed for dynamic bandwidth provisioning using WFQ vary by their traffic condition indicators, their objectives and their frequencies of update. A scheme called dynamic WFQ [13] adjusts weights based on the moving-averaged arrival rate to achieve a required delay differentiation, while another scheme called AWFQ [14] adjusts weights based on the queue length each time a new flow arrives and achieves better QoS performance than conventional WFQ. In reference [15], a scheme is described that constantly adjusts weights after each packet is served by using measured delay to alter the weights. This scheme requires tedious per packet update, and has the objective of maximizing bandwidth while achieving as low a delay as possible. All the above schemes however do not consider meeting specific QoS requirements.

More for DiffServ networks with WFQ implementation, [16] and [18] describe schemes that adaptively adjust WFQ weights on-the-fly according to changing traffic patterns, and are more in line with what we are trying to achieve. In reference [16], the authors propose a scheme for DiffServ networks that is based on a similar principle to Random Early Detection [17]. The average queue length is used to determine and adjust the weights periodically. By adjusting thresholds, the scheme is capable of adapting whenever the thresholds are exceeded. In reference [18], the authors propose a scheme that adjusts weights based on target traffic intensity as a target control value and measured traffic intensity as the feedback signal. The weights are adjusted whenever there is an overloaded or underloaded traffic condition. The delay requirement is met by limiting the queue length and the loss requirement is met by setting the target traffic intensity as a function of the loss requirement. For both schemes, the weight adjustment is based on control-feedback principle, which is at best a reactive mechanism. Hence, if the convergence time is longer than the traffic fluctuation cycles, the mechanism may not be able to reach steady state each time. Differently, our scheme tries to solve the bandwidth provisioning optimal control problem by using reinforcement learning. The method inherently has the capability to learn a policy that achieves long-term maximization, rather than short-term gain. In the next section, we describe how we use reinforcement learning in our RLAP scheme.

III. Continuous Space Reinforcement Learning

Reinforcement learning (RL) [19] is a machine learning theory that derives its roots from control principles. In RL, a learning agent has to formulate a policy, which determines the appropriate action to take in each state in order to maximize the expected cumulative reward over time. The reward is derived from how favorable the outcome is of the action taken by the agent in a particular state. In its original form, RL has been effectively used to solve problems with a discrete number of states and actions [20]. So far, only discrete-action-space RL methods have been applied to the area of communication networks [21-26]. However, most real-world problems (like the bandwidth provisioning problem) have a continuous state and action space (by this we mean that the state or action can take on real values in a range). As part of the REINFORCE algorithm [8] framework, Williams described a way to solve such problems using Gaussian units.

At each time step n, the Gaussian unit outputs real-valued actions through a Gaussian distribution function based on a mean μ_n and a variance σ_n, which are adjusted based on the evaluation of previous outcomes according to the following equations:

$$\mu_{n+1} = \mu_n + \alpha_\mu (r_n - \hat{r}_n)(y_n - \mu_n) \tag{2}$$

$$\sigma_{n+1} = \sigma_n + \alpha_\sigma (r_n - \hat{r}_n) \frac{(y_n - \mu_n)^2 - {\sigma_n}^2}{\sigma_n} \tag{3}$$

where r_n is the reward derived from the previous action y_n output from the Gaussian unit, $\hat{r}_n$ is the cumulative baseline reward, and α_μ and α_σ are appropriate adaptive rate constants for μ and σ. The cumulative baseline reward is maintained using an exponential averaging scheme:

$$\hat{r}_n = \gamma r_n + (1 - \gamma)\hat{r}_{n-1} \tag{4}$$

where $0 < \gamma \leq 1$. $\hat{r}_n$ effectively provides a basis of evaluation for each new outcome caused by the previous action.

The idea behind the algorithm is to add a perturbation (the difference between the output value y_n and the mean μ_n), and observe whether this perturbation has caused the unit to receive an evaluation signal that is more than the expected evaluation. If so, it would be desirable for the unit to produce an output closer to the current output. The mean output value should therefore move in the direction of the perturbation. If the evaluation received is less than expected, then the opposite should happen, i.e., the unit should adjust its mean in the opposite direction. This is effectively what equation (2) does. Using equation (3), the amount of exploration σ_n of the Gaussian unit is controlled by narrowing the search if an improvement is found near the current mean or when it is penalized for an exploration far from the current mean, and vice versa. The exploration would continue till the point when the optimal is reached and σ becomes zero. The algorithm converges to the optimal value via a combination of a gradient-following strategy and exploration control. By using reward feedback, the algorithm is able to maximize long-term reward through reward-optimized control.

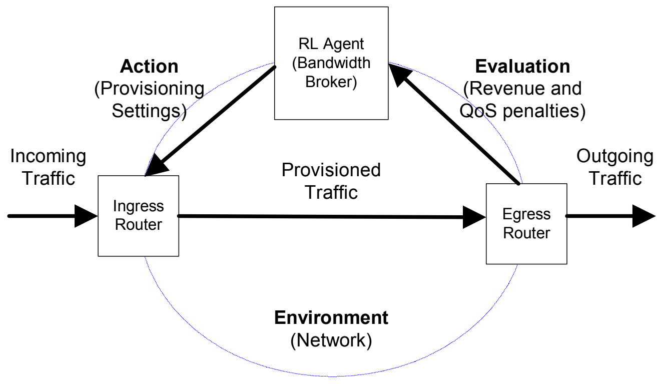

Figure 1. Reinforcement Learning Loop in RLAP

IV. Reinforcement Learning-based Adaptive Provisioning (RLAP)

A. Motivation

The motivation behind the proposed Reinforcement Learning-based Adaptive Provisioning (RLAP) scheme is that SLAs tied to a usage-based pricing plan can be used as a feedback mechanism to determine a policy, which adaptively adjusts PHB aggregate bandwidth provisioning to achieve long-term revenue maximization. Often, SLAs are drawn, based on a pricing plan, between providers and users independent of the current network configuration. Network providers would then have to provision their networks (via router configuration) based on these SLAs. However, up till now, network configuration is still an open problem.

To this end, we propose to use RL, which is able to maximize a long-term objective reward function, to solve the bandwidth provisioning problem. The reward function is constructed to be the amount of net revenue earned based on the SLA and pricing plan contracted. The RL agent adaptively adjusts the bandwidth provisioning at a regular interval based on the feedback of how much revenue was generated in the last interval. Since the provisioning based on current traffic conditions affects the QoS experienced and the revenue earned, the action-reward forms a closed-loop as illustrated in Figure 1.

There are good reasons for using RL in our scheme. Firstly, RL does not require the complex analysis of the underlying system to determine the correct actions to take. It only takes into account how well the system is performing relative to the state and the action taken. Secondly, RL makes use of simple statistical hill-climbing updates to converge to a policy over time through experience, requiring little computational resources. The policy learned may at times be superior to analytically-determined policies. Thirdly, RL systematically assigns credit to a multi-agent system, as is our case, where there are many routers involved.

We foresee that RLAP can be used in the following scenarios to improve provisioning:

1. Changing traffic conditions – due to changing traffic intensities of EF and AF traffic, EF and AF weights need to be adjusted accordingly. Under-provisioning may lead to high delays and low throughput. On the other hand, over-provisioning may not be necessary when traffic intensity is low.
2. Different pricing plans – by changing the reward function, weights can be changed in favor of the traffic that generates more revenue. If penalties are not severe, weights can be lowered to take advantage of revenue from other traffic, allowing the occasional penalties.
3. Strictness of SLS – depending on the delay and throughput bounds and the level of congestion in the network, weights can be changed to reflect how critical these bounds are. The lower the bound and the higher the traffic intensity, the greater should be the weight.

Our choice of a continuous-space Gaussian unit for bandwidth provisioning becomes apparent when we consider that the WFQ weight settings that the router has to control to adjust provisioning are continuous parameters. An additional advantage is that Gaussian units implemented for each router only require a minimal amount of memory space to store the μ and σ values, unlike its discrete-space RL counterparts, which often require large memory spaces to store look-up tables [27].

B. RLAP Framework

To describe how RLAP can be implemented in a DiffServ network, a one-domain topology in Figure 2 is used in this paper as an example, and is made as simple as possible without loss of generality. Sources S_0 to S_7 have destinations D_0 to D_7 respectively.

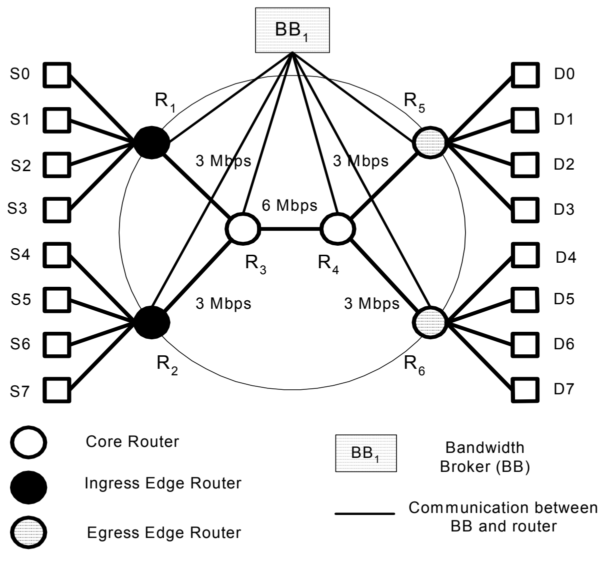

Figure 2. DiffServ Network Topolgy

A bandwidth broker BB_1 is used as the centralized collection and decision-making point. The framework we use is similar to that used in [28], putting RLAP in place instead for determining bandwidth provisioning. RL agents can either be placed in each router, or housed in the bandwidth broker with separate logical agents for each router. We chose the latter, to be in line with the framework. This design also requires almost no modification to existing routers to implement RLAP based on the framework. At regular intervals, BB_1 collects traffic measurements (in terms of number of bits) from all routers in the domain for computation of revenue.

Concurrently, destination nodes report, via a system of accounting, any QoS violations in terms of amount of traffic delayed and number of intervals throughput was not met, to the bandwidth brokers along the paths of the flows (in a multi-domain case); BB_1 in our topology.

BB_1 then makes decisions through the RL agents and sends the WFQ weight configurations to the respective routers. SLAs and pricing plans are stored in the database for computing the charges and penalties used as feedback to the RL agents. In a multi-domain scenario, bandwidth brokers may shift some of the functions up the hierarchy of bandwidth brokers. In return, decisions made are passed downwards towards the routers.

C. REINFORCE for RLAP

In implementing REINFORCE Gaussian units in the RLAP scheme, we have set the output y of the Gaussian unit (henceforth referred simply to as an RL agent in this paper) to be the WFQ weight settings that adjust provisioning. The time step n is set to be the n^{th} time interval since the RL agent started; each time interval being of duration T. At the end of each interval, the cumulated reward r, based on the evaluation of the weight settings for traffic conditions over the interval, would be used as feedback to adaptively adjust μ and σ of the RL agent, which are used to determine the next weight settings. For RL agent RL_i, y_i, μ_i and σ_i are vectors with elements $y_{i,j}$, $\mu_{i,j}$ and $\sigma_{i,j}$ for each PHB aggregate j.

For each agent RL_i, the full RLAP algorithm is as follows (omitting the subscript i for clarity):

```
Initialize $\mu_0$ and $\sigma_0$.
At the end of every interval $n$,                $n \geq 1$
update        $\hat{r}_n = \gamma r_n + (1-\gamma)\hat{r}_{n-1}$
              $\mu_{n+1} = \mu_n + \alpha_\mu (r_n - \hat{r}_n)(y_n - \mu_n)$
and           $\sigma_{n+1} = \sigma_n + \alpha_\sigma (r_n - \hat{r}_n)\frac{(y_n - \mu_n)^2 - {\sigma_n}^2}{\sigma_n}$
Select        $y_n \sim N(\mu_n, \sigma_n)$
```

The choice of initial values for μ_0 and σ_0 is an important issue in these types of REINFORCE algorithms, such as ours. The effect of the choice of different initial values for μ_0 is shown and discussed in our simulations in Section VI.A. For all real practical cases, we suggest that the choice of μ_0 be based on any limited *a priori* information available; for example, traffic specifications given by the user and past traffic measurements may provide average and peak throughput information, which can be used as the initial provisioning values.

The choice of σ_0 was made such that a trade-off was struck between the smoothness of the exploration and the rate of convergence. Larger σ_0 values tend to provide better convergence to the global optimal solution, however, the fluctuations in μ values may be great and undesirable. Smoothness of exploration is important during the real operation of the network. A possible way to solve this concern is to have a trial phase, where a large σ_0 may be used to train the RL agents, before actual (chargeable) network traffic is carried. We have not done this in our experiments and σ_0 was set to a rather low value. This was done intentionally to demonstrate the capability of RLAP under situations where such trial periods are not possible, and large fluctuations in network settings are undesirable.

The selection of values for constants α_μ, α_σ and γ, will be discussed later in Section VII as part of our discussion on implementation issues of RLAP.

D. Reward Function

To determine the feedback reward r, we propose a 3-tier pricing plan as the evaluation function. Users are charged based on the amount of traffic carried on the network. As such, users pay only for what they use. A different price is charged for each PHB of traffic and a penalty is imposed for each SLS not met and for each packet loss. EF traffic has a penalty for delay and AF traffic has both delay and throughput penalties. The rationale behind the use of such a usage-based pricing plan is that users often do not know their traffic specifications *a priori*, or do not want to be committed to them.

Charging them based on usage, may offer a better alternative [29] than the commonly-used leased-line approach that charges based on the bandwidth specified for a certain period of time, regardless of the actual utilization of the bandwidth. Coupled with the usage-based pricing, the provider refunds QoS penalties based on the amount of traffic that breeched the service level, instead of strictly guaranteeing a service level for all traffic and paying a fixed penalty. This penalty-based approach acts as a disincentive for providers to breech the service level contracted, whilst leaving leeway for occasional lapses, which the end-user may not mind as long as he is adequately compensated; for example, at a rate of more than double the charge of transmission. This pricing plan is attractive to both providers and users as it encourages increased network utilization, leading to higher revenues, as well as greater savings, due to users being able to pay for only what they use.

The reward evaluation function we propose to use computes the reward r_i for RL agent RL_i in the following way:

$$r_i = \sum_j \begin{pmatrix} c_j \times t_{i,j} - p_{loss,j} \times l_{i,j} - p_{dly,j} \times d_{i,j} \\ - p_{thr,j} \times th_{i,j} \end{pmatrix} \qquad (5)$$

For each PHB aggregate j, c_j is the charge per bit of traffic carried, $t_{i,j}$ is the amount of traffic forwarded by router i, $l_{i,j}$ is the number of packets loss at router i, $d_{i,j}$ is the amount of traffic that passed through router i not meeting delay QoS and $th_{i,j}$ is the number of intervals not meeting throughput QoS for traffic that passed through router i. $p_{loss,j}$ is the penalty per packet loss, $p_{dly,j}$ is the penalty per bit of PHB aggregate j not meeting delay QoS and $p_{thr,j}$ is the penalty per interval of PHB aggregate j not meeting throughput QoS.

V. NS-2 Implementation

A. NS-2 Setup

Using *ns-2* [8] DiffServ extensions, the topology in Figure 2 was set up to compare RLAP with static provisioning. We show that RLAP is able to improve even over commonly used over-provisioning methods (this requires the assumption that users are able to describe their usage, which is not always possible). Droptail queues were used for each class instead of the usual RIO (RED with In and Out packets) buffer management to separate the effects of RLAP from RIO. The buffer length for AF and BE traffic were set to 100, while the buffer length for EF traffic was set to 2 for the static provisioning case (as is the common practice to keep end-to-end delay low) and 10 for our scheme. We show that RLAP is able to accommodate more packets in the buffer, and yet not hamper end-to-end delay by much.

B. Traffic Characteristics

In our simulations, traffic from all 3 PHB aggregates were generated on all links in the DiffServ domain. The links were made to carry close to full capacity most of the time. Sources S_0 to S_3 have similar characteristics to sources S_4 to S_7 respectively. The characteristics of sources S_0 to S_3 are given in that order in Table I.

TABLE I.

CHARACTERISTICS OF TRAFFIC SOURCES

Source	**Traffic Type**	**Inter-arrival Time (s)**	**Holding Time (s)**	**ON Rate (kbps)**
S_0, S_4	EF	1.875	30	64
S_1, S_5	BE	7.5	60	128
S_2, S_6	AF	4.5	30	300
S_3, S_7	BE	1.775	30	128

S_0 is an EF source that represents delay bounded traffic like VoIP traffic. S_2 is an AF source that represents loosely delay bounded, high throughput requirement, traffic like video traffic. Both are modeled as exponential ON-OFF sources with same ON (500ms) and OFF (500ms) times. S_3 is a BE source that represents non-bounded aggregated web traffic. It is modeled as a pareto ON-OFF source with the same ON and OFF times as the EF and AF traffic. These 3 sources run over UDP. The last source S_1 runs over TCP, and is a BE source that represents non-bounded high throughput traffic like FTP traffic. It is a CBR source that consumes any unused capacity on the link. This enables the link to be fully utilized most of the time. Sources S_0 to S_7 have destinations D_0 to D_7 respectively. The average amount of generated traffic towards each ingress router for EF and AF are 500 kbps and 1 Mbps respectively. BE traffic utilizes the remaining amount of capacity; about 1.5 Mbps.

C. Experimental Details

In the static provisioning case, we set up the WFQ weights to be static throughout each experiment lasting 20,000s. For the RLAP case, we set up the experiment to initially begin with the same weights as in the static case. We also set μ_0 to be equal to the static setting. The RLAP scheme then kicks in after 1,500s, adjusting the WFQ weights in the routers adaptively and improving the policy in the RL agents through time. All simulations involving RLAP were run with 5 different random seeds. The mean and range is plotted wherever possible. For both cases, we measure performance only after an initial period of 5,000s for an additional 15,000s. The time-step interval T chosen is 500s. The choice of α_μ, α_σ and γ used are 0.00005, 0.00001 and 0.2 respectively. The effect of the choice of these parameter values will be discussed in Section VII.

Table II shows the pricing plan used. The charge for EF and AF traffic is 10 and 4 times the charge for BE traffic respectively. The penalties for packet losses and delayed EF traffic are double the charge. The penalties for AF traffic not meeting the delay or the throughput bound is equal to its charge, such that if AF traffic does not meet both requirements, it will be penalized double its charge. The delay requirements for EF and AF are 15 ms and 35 ms respectively, and the throughput requirement for AF is 200 kbps.

TABLE II.

PRICING PLAN

c_{EF}	0.0001	$p_{loss,EF}$	0.2	$p_{dly,EF}$	0.2
c_{AF}	0.00004	$p_{loss,AF}$	0.08	$p_{dly,AF}$	0.04
c_{BE}	0.00001	$p_{loss,BE}$	0.02	$p_{thr,AF}$	10

TABLE III.

WFQ WEIGHT SETTINGS FOR VARIOUS PROVISIONING STRATEGIES

Provisioning Strategy	**WFQ Weight Settings (EF:AF:BE)**
Under-provisioning	1:2:3
50% over-provisioning	3:4:5
Over-provisioning	1:1:1

VI. SIMULATION RESULTS

A. Comparison under Different Initial Provisioning

In our first experiment, we compare how RLAP improves over various initial WFQ weight settings. Since RLAP only kicks in after 1,500s, the weights remain static for the initial period. After which, RLAP adjusts the weights from these initial values. We benchmarked RLAP against static provisioning, which maintains the WFQ weights throughout the experiment. For static provisioning, the different WFQ weight settings mean different provisioning strategies. Three strategies were tested. The under-provisioning strategy provisioned EF at the expected average traffic rate, i.e., 0.5 Mbps of bandwidth. The 50% over-provisioning strategy allocated 50% above the average EF traffic rate, i.e., 0.75 Mbps of bandwidth. The most commonly-used strategy, the over-provisioning strategy, which allocates EF with more than sufficient bandwidth to handle bursts, provisions EF traffic at twice the bandwidth. For all 3 strategies, AF traffic was provisioned at the expected average AF traffic rate, i.e. 1.0 Mbps, while BE was allocated the remaining of the capacity. It is to be noted that static provisioning settings can only be determined when the average or peak rate information is accurately available *a priori*. On the other hand, RLAP requires only an estimate as an initial point. For our experiment, the average rate of the traffic generated by simulation is assumed to be known in order to have a comparison. The initial weight settings are given in Table III.

Figures 3 and 4 show the average throughput comparisons and figures 5 and 6 show the average delay comparisons across flows in each PHB for the various strategies. It can be seen that RLAP is able to find a policy that improves QoS for AF traffic and maintains QoS for EF traffic at the expense of QoS for BE traffic. This is despite having increased throughput for EF and AF traffic. We also observe that RLAP is able to adjust to this policy regardless of the initial weight settings. For the case of static provisioning, we see that as the initial level of EF provisioning increases, the average throughput and delay of each PHB aggregate changes. This shows the level of bias given to provisioning EF. The performance for RLAP however, is almost consistent. This clearly demonstrates the ability of the algorithm to find an optimum strategy regardless of the initial values used.

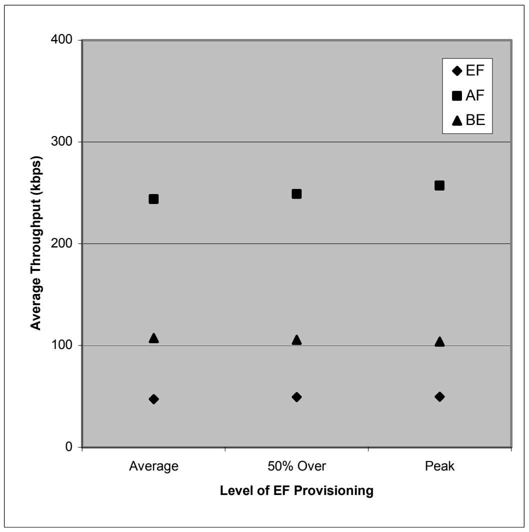

Figure 3. Average throughput per flow under different initial provisioning for Static Provisioning

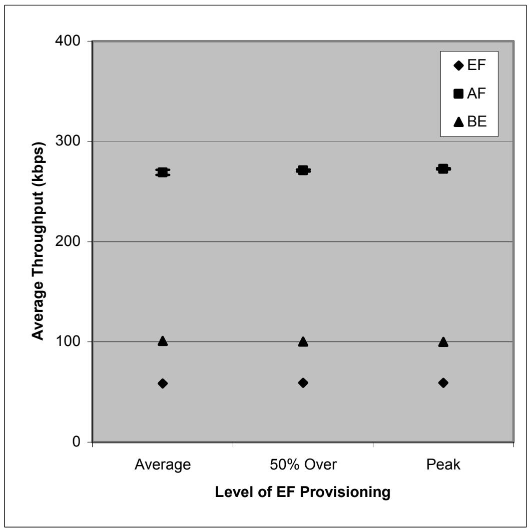

Figure 4. Average throughput per flow under different initial provisioning for RLAP

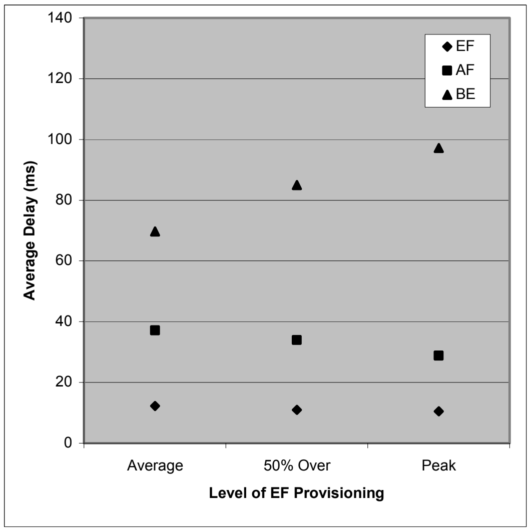

Figure 5. Average delay under different initial provisioning for Static Provisioning

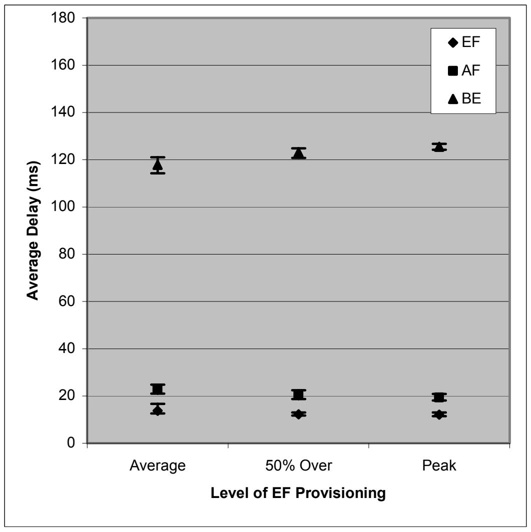

Figure 6. Average delay under different initial provisioning for RLAP

Figure 7 shows a chart of the improvement in revenue that RLAP makes over static provisioning. This is a key feature of RLAP; the ability to adapt provisioning based on a pricing plan and QoS requirements, such that long-term revenue is maximized. The slight difference in levels of revenue for the RLAP case shows that though RLAP improves over static provisioning, the convergence to the optimal policy is slightly slower for the average and 50% over-provisioning cases. As with all adaptive algorithms, this is due to the need to converge from an initial point further away from the optimal point. Nonetheless, convergence is still achieved.

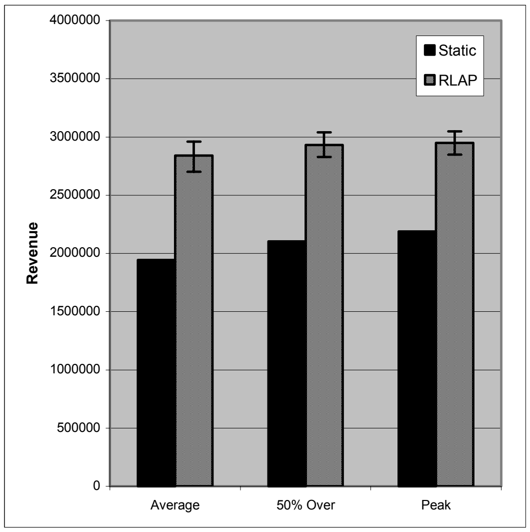

Figure 7. Revenue comparison under different initial provisioning

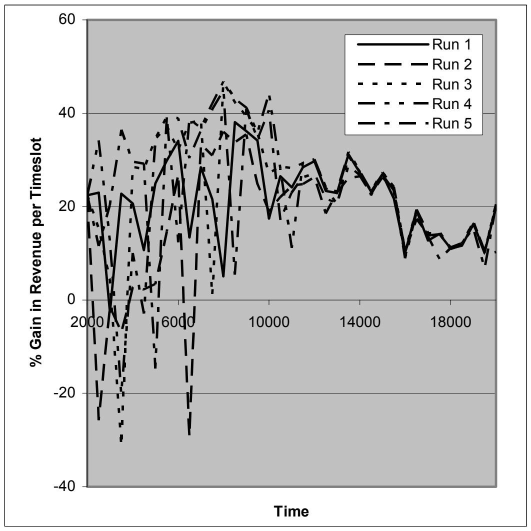

Figure 8. Percentage gain in revenue of RLAP over static provisioning across time

TABLE IV.

QOS AND REVENUE ACHIEVED FOR STATIC PROVISIONING

Traffic Type	Throughput (bps)	Delay (ms)	Revenue
EF	51,679	10.5	804,743
AF	261,006	26.3	838,167
BE	105,262	87.6	464,828

TABLE V.

QOS AND REVENUE ACHIEVED FOR RLAP

Traffic Type	Throughput (bps)	Delay (ms)	Revenue
EF	59,171	12.2	1,147,587
AF	273,094	18.7	1,159,049
BE	103,048	106.3	439,558

B. Comparison under Changing Traffic Conditions

In the second experiment, we set out to compare how RLAP improves over static provisioning over varying traffic conditions. EF traffic from S_0 and S_4 was halved after 10,000s and 15,000s respectively to cause the change. We used the peak-rate over-provisioning strategy for this experiment and the rest that follow since it is the common practice. Tables IV and V summarize the results obtained and figure 8 shows the gain in revenue of RLAP over static provisioning across time. We see that RLAP always performs better than the static provisioning case and that it is able to adapt well to the changes in traffic pattern at time 10,000s and 15,000s, evidenced by a drop followed by an increasing trend in gain in revenue. Increase in gain is observed despite more favorable conditions for the static provisioning (since there is relatively lighter priority traffic). We also see that the QoS performance is similarly good, as in the previous experiment. By plotting out all 5 runs in figure 8, we see a trend that regardless of the random seed used, the algorithm would still converge. During the initial period, there is high fluctuation due to learning and exploration. But after 10,000s, all 5 runs converge. This trend is seen in other experiments as well.

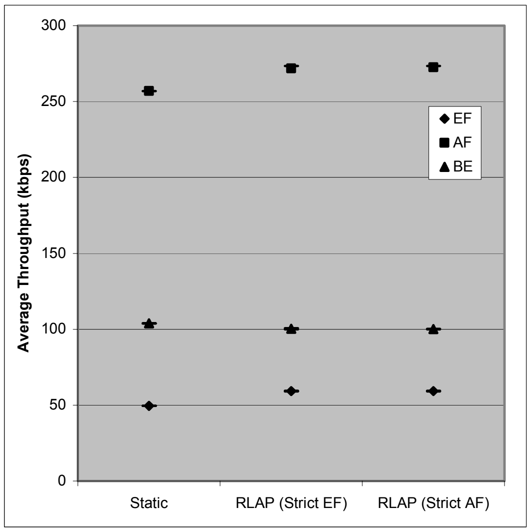

Figure 9. Average throughput comparison under different QoS requirements

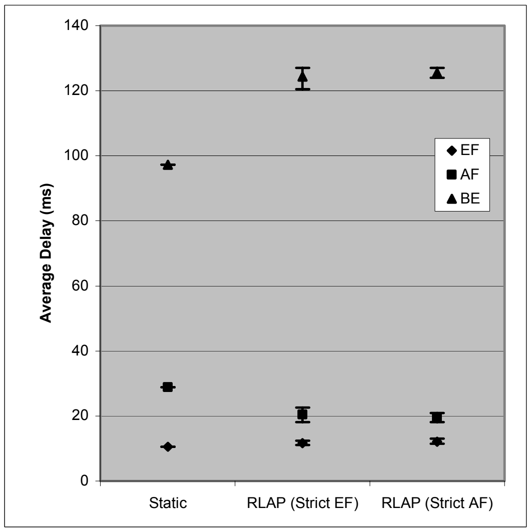

Figure 10. Average delay comparison under different QoS requirements

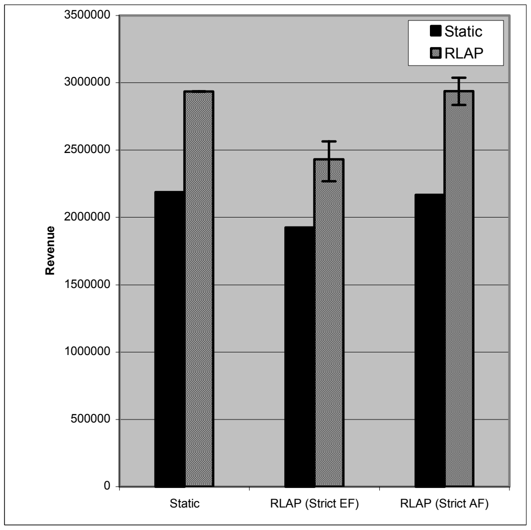

Figure 11. Revenue comparison under different QoS requirements

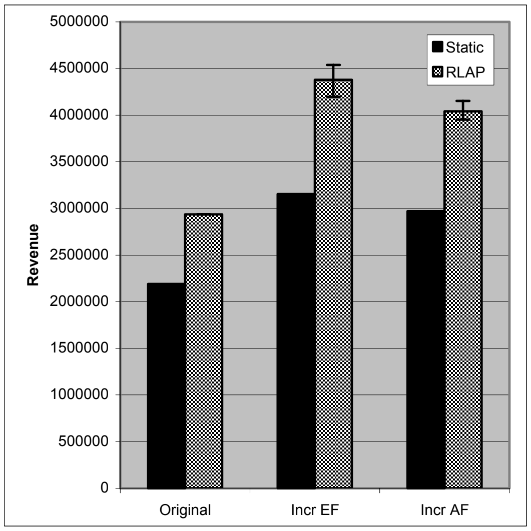

Figure 12. Revenue comparison under different pricing plans

C. Comparison under Different QoS Requirements

In our third experiment, we seek to demonstrate how RLAP performs when QoS requirements are changed. To alter the QoS requirements, we reduced the EF delay bound to 12ms in one case and the AF delay bound to 30ms in the other case to represent stricter EF and AF requirements respectively. Figures 9 and 10 show a comparison of average throughput per flow and average delay across flows for each PHB between static provisioning and RLAP for both QoS cases. As the results do not change with QoS settings for the static provisioning case, the values for static provisioning in the figure are the same for the stricter EF and the stricter AF cases. Hence, we classify the static provisioning cases together under "static". Note that the range markers may not be clear in each of the figures. This is attributed to the small variation in performance between the 5 runs.

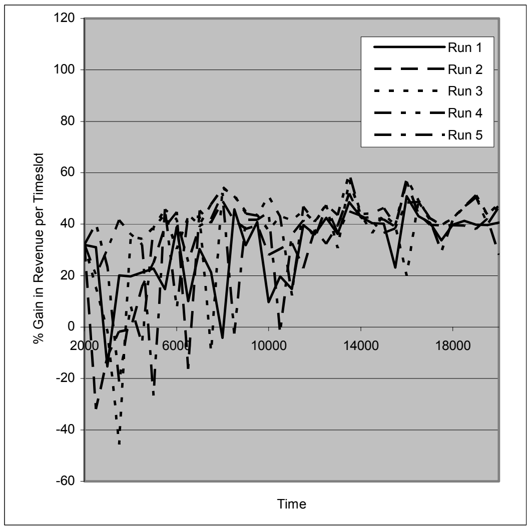

Figure 13. Percentage gain in revenue across time for increased EF pricing plan

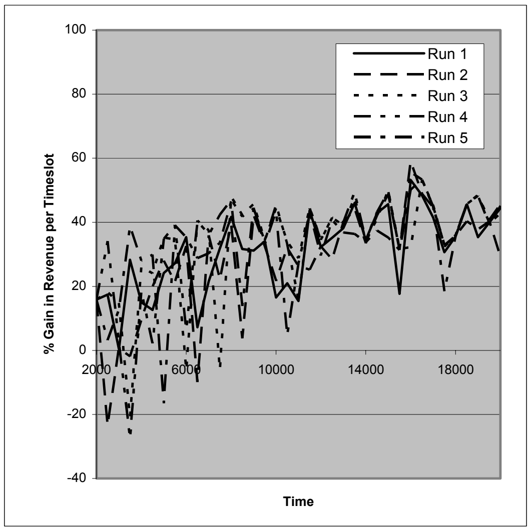

Figure 14. Percentage gain in revenue across time for increased AF pricing plan

We see that RLAP learns a policy that improves the performance of AF QoS as AF QoS requirements becomes stricter. We recall that AF is provisioned at the average rate initially. This is clearly not sufficient in this case, and under-provisioned AF traffic gets penalized heavily. As such, the effect of QoS requirements must be taken into account. It is also noted that in the case of stricter EF QoS requirements, RLAP does not see the need to improve the performance of EF QoS, due to the already good QoS performance. This means that not all adjustments to QoS requirements need to be taken into account equally. RLAP demonstrates here its capability to act accordingly. Figure 11 shows the consequent improvements in revenue earned as a result of RLAP.

D. Comparison under Different Pricing Plans

In the final experiment, we seek to confirm that RLAP performs according to the pricing plan used; unlike most provisioning schemes proposed that do not consider this. To vary the pricing plan, we doubled EF revenue and penalties for one case and doubled AF revenue and penalties in the other case. Figure 12 shows the improvement in revenue made between static provisioning and RLAP.

We see that RLAP makes more significant improvements in net revenue earned than static provisioning when the pricing of EF and AF traffic carried is doubled. This shows clearly that RLAP adjusts weights according to the pricing plan as well. A larger gain in profits will encourage the RLAP scheme to increase provision of that component which is causing the good result.

Figures 13 and 14 show the percentage gain in revenue across time of RLAP in comparison to static provisioning. We see that RLAP initially requires a learning phase where at times it fairs worse than the static provisioning. However, after about 10,000s, RLAP begins to make significant improvements. In both figures, we see that there is a convergence. The fluctuations observed towards the latter part of the simulation are not due to non-convergence, as confirmed by decreasing σ values. Rather, they are due to fluctuating traffic conditions.

From all the results of the above experiments, one might be tempted to think that if we set out to provision EF and AF traffic at higher rates initially and left them static, we could achieve the same results as RLAP. Unfortunately, we could not have known how much to provision a priori, and if traffic conditions were to change or pricing plans and QoS requirements altered, we would not be able to determine the corresponding provisioning without complex analysis, which may require certain assumptions. This is where RLAP provides a simple solution that is able to learn and adapt without supervision or expert analysis.

VII. Discussion On Implementation Issues

In any proposal that implements RL agents, convergence is always an important issue. In continuous RL methods, the assumption that the reward function has no discontinuity in the practical range and also be monotonic towards the global optimum must hold [27]. If not, such algorithms may be stuck in a local maximum. In our case, the pricing plan must have a monotonically increasing form, which is usually the case.

A related issue is the rate of convergence. In RLAP, this is controlled by the constants α_μ, α_σ and γ. By adjusting their values, a trade-off is made between the rate of convergence and the fluctuation in action values.

Another parameter to consider is time interval T. It should be set up to balance the trade-off between the adaptability of the scheme and the frequency of disruption caused by network re-configurations. We prefer a larger interval as it causes less disruption to the network and also allows RLAP to be less susceptible to traffic irregularities, due to the averaging of traffic and reward/penalty measurements over longer periods.

VIII. Conclusion

In this paper, a Reinforcement Learning-based Adaptive Provisioning (RLAP) scheme has been proposed, together with detailed design and implementation. A continuous reinforcement learning method has been chosen to solve the resource management problem of bandwidth provisioning, due to its ability to learn a policy that matches states to optimal actions. The novel use of a usage-based pricing policy coupled with a service-level agreement (SLA) as a means of feedback to the RL agent, enables RLAP to maximize long-term revenue. We have shown through simulations that RLAP is able to adapt well to changing traffic conditions, as well as various pricing plans and QoS specifications.

Being the pioneers to use a continuous RL method in network resource management, it is our intention for RLAP to become a model for future works in this area. In our future work, we plan to use RLAP in a multi-domain network. We are also in the midst of using continuous RL methods to optimally configure other network parameters.

References

[1] IETF Differentiated Services (DiffServ) Working Group, *http://www.ietf.org/html.charters/diffserv-charter.html*

[2] S. Blake, et al, "An Architecture for Differentiated Services", *IETF RFC 2475,* Dec 1998.

[3] V. Jacobsen, et al, "An Expedited Forwarding PHB", *IETF RFC 2598,* Jun 1999.

[4] J. Heinanen, et al, "Assured Forwarding PHB Group", *IETF RFC 2597,* Jun 1999.

[5] D. Goderis, et al, "Service Level Specification Semantics, Parameters and negotiation requirements", *IETF Internet Draft,* Jun 2001.

[6] K. Nichols, V, Jacobsen, and L. Zhang, "An Approach to Service Allocation in the Internet", *IETF Internet Draft,* Nov 1997.

[7] D. Clark, and W. Fang, "Explicit allocation of best-effort packet delivery service", *IEEE/ACM Trans. on Networking,* 6(4), pp. 362-373, 1998.

[8] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning", *Machine Learning,* 8(3), pp. 229-256, 1992.

[9] S. McCanne, and S. Floyd, *ns-2 – The Network Simulator,* available from http://www.isi.edu/nsnam/ns/.

[10] A. Demars, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", *Proc. of ACM SIGCOMM '89*, Sep 1989.

[11] S. J. Golestani, "A Self-Clocked Fair Queuing Scheme for Broadband Applications", *Proc. of IEEE INFOCOM '94,* Jun 1994.

[12] J. C. R. Bennet, and H. Zhang, "WF^{2}Q: Worst-Case Fair Weighted Fair Queuing", *Proc. of IEEE INFOCOM '96,* 1996.

[13] C-C. Li, et al, "Proportional Delay Differentiation Service Based on Weighted Fair Queuing", *Proc. IEEE ICCCN 2000,* Oct 2000.

[14] M-F. Horng, et al, "An Adaptive Approach to Weighted Fair Queue with QoS Enhanced on IP Network", *Proc. IEEE TENCON 2001,* Aug 2001.

[15] D. Hang, et al, "TD^{2}FQ: An Integrated Traffic Scheduling and Shaping Scheme for DiffServ Networks", *Proc. of IEEE HPSR 2001,* May 2001.

[16] H. Wang, C. Shen, and K. G. Shin, "Adaptive-Weighted Packet Scheduling for Premium Service", *Proc. of IEEE ICC 2001,* Jun 2001.

[17] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Trans. Networking,* 1(4), 1993, pp. 397-413, 1993.

[18] R-F. Liao, and A. T. Campbell, "Dynamic Core Provisioning for Quantitative Differentiated Service", *Proc. of IEEE IWQoS 2001,* Jun 2001.

[19] A. Barto, R. Sutton, and C. Anderson, "Neuron-like Elements that can Solve Difficult Learning Control Problems", *IEEE Trans. on Systems, Man and Cybernetics,* vol. 13, pp. 835-846, 1983.

[20] C. J. C. H. Watkins, and P. Dayan, "Q-Learning", *Machine Learning,* vol. 8, pp. 279-292, 1992.

[21] A. Vasilakos, M. Saltouros, A. Atlasis, and W. Pedrycz, "Optimizing QoS Hierarchical routing in ATM nets using Computational Intelligence", *IEEE Trans. on Systems, Man and Cybernetics,* Special Issue on Computational Intelligence, Openness and Programmability in Networks and Internet Services, Aug 2003.

[22] A. Atlasis, and A. Vasilakos, "LB-SELA: Rate-based access protocol for ATM networks", *Computer Networks and ISDN Systems*, 30 (1998), pp. 963-980, May 1998.

[23] A. Atlasis, N. Loukas, and A. Vasilakos, "The use of learning algorithms in ATM networks CAC problem: a methodology", *Computer Networks and ISDN Systems*, 34(2000), pp. 341-353, Sep 2000.

[24] H. Tong and T. X. Brown, "Adaptive Call Admission Control under Quality of Service Constraints: A Reinforcement Learning Solution", *IEEE JSAC,* 18(2), Feb 2000.

[25] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, "Call Admission Control and Routing in Integrated Services Networks using Neuro-Dynamic Programming", *IEEE JSAC,* 18(2), Feb 2000.

[26] C-K. Tham, and Y. Liu, "Minimizing Transmission Costs through Adaptive Marking in Differentiated Services Networks", *IEEE MMNS 2002,* Oct 2002.

[27] G. A. Rummery, "Problem Solving with Reinforcement Learning", PhD Thesis, University of Cambridge, Cambridge, England, Jul 1995.

[28] C. N. Chuah, L. Subramanian, R. H. Katz, and A. D. Joseph, "QoS Provisioning using a Clearing House Architecture", *Proc. of IEEE IWQoS 2000,* Jun 2000.

[29] L. A. DaSilva, "Pricing for QoS-Enabled Networks: A Survey", *IEEE Communication Surveys and Tutorials*, 3(2), pp. 2-8, 2000.