Market-Based Hierarchical Resource Management using Machine Learning

Ramy Farha and Alberto Leon-Garcia

University of Toronto, Toronto, Ontario, Canada
ramy.farha@utoronto.ca, alberto.leongarcia@utoronto.ca

Abstract. Service providers are constantly seeking ways to reduce the costs incurred in managing the services they deliver. With the increased distribution and virtualization of resources in the next generation network infrastructure, novel resource management approaches are sought for effective service delivery. In this paper, we propose a market-based hierarchical resource management mechanism using Machine Learning, which consists of a negotiation phase where customers are allocated the resources needed by their activated service instances, and a learning phase where service providers adjust the prices of their resources in order to steer the network infrastructure towards the desired goal of increasing their revenues, while delivering the mix of services requested by their customers. We present the operation of such a market where distributed and virtualized resources are traded as commodities between autonomic resource brokers performing the negotiation and learning on behalf of service providers. We perform extensive simulations to study the performance of the proposed hierarchical resource management mechanism.

1 Introduction

Service providers (SPs) are reacting to competitive pressures by transitioning from being providers of individual services (voice, data, video) to providers of service bundles. SPs must deal the challenge of reducing the costs of managing these services and the network infrastructure over which these services are deployed and offered. The requirements for a powerful service management system motivate the need to automate management by evolving to self-managing infrastructures, in order to ensure automated service delivery to customers.

The Autonomic Computing [1] concept presented by IBM to reduce software complexity and cost of service delivery in the IT domain is attractive in the sense that a similar concept could be mapped in the telecommunications domain to perform autonomic service management. In a previous work [2], we had introduced the Autonomic Service Architecture (ASA), which aims to give SPs the needed solutions to dynamically marshal their service delivery infrastructure and to support the required service mix at a given point in time using a hierarchy of Autonomic Resource Brokers (ARBs) which perform resource management.

The network infrastructure for telecommunications service delivery could be viewed as a set of competing SPs, similar to players in a game-theoretic problem [3]. Given the difficulty to represent this game using classical game theory, we explore an alternative approach using Machine Learning [4], where Reinforcement Learning [5] agents built into the ARBs of ASA incrementally improve their strategies according to trial-and-error interactions with the external world. As a result, SPs adjust the prices of their resources to achieve better performance for the autonomic resource management approach performed by ASA.

In this paper, we present a market-based hierarchical resource management approach, using Machine Learning to autonomically steer the network infrastructure towards the desired goals for both SPs and their customers. The remainder of this paper is structured as follows. In section 2, we review some related work. In section 3, we summarize the design of ASA, prior to explaining the hierarchical resource management approach in section 4. In section 5, we integrate Machine Learning into ASA to improve the strategy of SPs. In section 6, we illustrate some simulation results to show the performance of the proposed scheme. Finally, in section 7, we conclude the paper and suggest some future work.

2 Related Work

Game Theory is aimed at understanding situations in which several decision makers (also called players) interact [3]. Classical game theory suffers from several shortcomings, such as the need for perfect information about strategies of other players, or about the probability distribution of their strategies. Such assumptions are infeasible in an environment of competing service providers (SPs), which are the players in the game-theoretic problem, hence the need for alternative ways to solve this game, such as Machine Learning [4].

Machine Learning is a branch of artificial intelligence that encompasses areas such as Neural Networks, Genetic Algorithms, Ant Colony Optimization, and Reinforcement Learning (RL) [5]. RL will be used to enable the decision makers, in this case the competing SPs, to optimize their operation using trial-and-error interactions with the external environment, in order to steer the operation of the network infrastructure towards the greater good for both themselves and for their customers, without the need to share any information with other SPs.

The closest work to this paper is presented by Wang and Li [6]. In their approach, concepts from control theory are used to help selfish nodes in a service overlay network incrementally adapt to the market, by making optimized strategic decisions based on past experiences. While the techniques they use are comparable to those adopted in this paper, their application is drastically different. This paper involves a different negotiation mechanism since some customers specify the rate required instead of only having a Best Effort service as is the case in the work of Wang and Li. The complexity of the problem changes as well since this paper uses a hierarchical architecture. Furthermore, this paper attempts to improve the utilization of the network infrastructure's virtual resources by SPs while satisfying customer requirements, whereas Wang and Li are attempting to force selfish nodes in a service overlay network towards more cooperation.

3 Autonomic Service Architecture Overview

The main task of the Autonomic Service Architecture (ASA) is to automate the delivery of services offered by a service provider (SP) to its customers in next generation networks. ASA achieves this goal through the interaction of self-managing entities, called Autonomic Resource Brokers (ARBs), which autonomically handle provisioning, management, and termination of offered services.

The layered structure of ARBs in ASA is shown in Fig. 1. When customers activate service instances they have bought from SPs, these service instances are managed by the SPs using Service Instance ARBs (SIARBs). The multiple service instances of a particular service offered by a SP are managed by Composite ARBs (CARBs). The different services offered by a SP (managed by CARBs) are managed by a Global ARB (GARB), which handles all the resources available at this SP's disposal. Physical resources are virtualized into virtual resources to deal with heterogeneity, using concepts similar to those in [7].

Resource management in next generation networks using ASA could be seen as a market where virtual resources are exchanged between customers and SPs. This view is becoming a reality with architectures such as the one used by Amazon [8], or with the proliferation of Grids [9]. Virtual resources can be assimilated to commodities with prices that vary depending on the demand. The market consists of several competing SPs owning the different virtual resources and services which are offered to customers. In the upcoming sections, we elaborate on how the market model applies for hierarchical resource management in ASA.

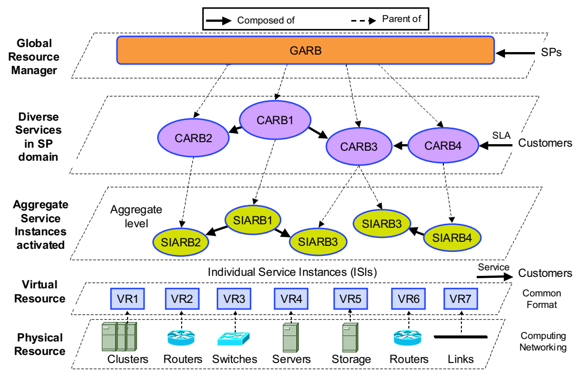

Fig. 1. Autonomic Resource Brokers Hierarchy in the Autonomic Service Architecture

4 Hierarchical Resource Management Algorithm

For each virtual resource in the network infrastructure, we run the proposed market-based hierarchical resource management algorithm with Machine Learning, according to the autonomic loop shown in Fig. 2. This algorithm, which will be detailed next, is performed by the Autonomic Resource Brokers (ARBs) of the aforementioned Autonomic Service Architecture (ASA). The notations used in the rest of this paper are shown in Table 1.

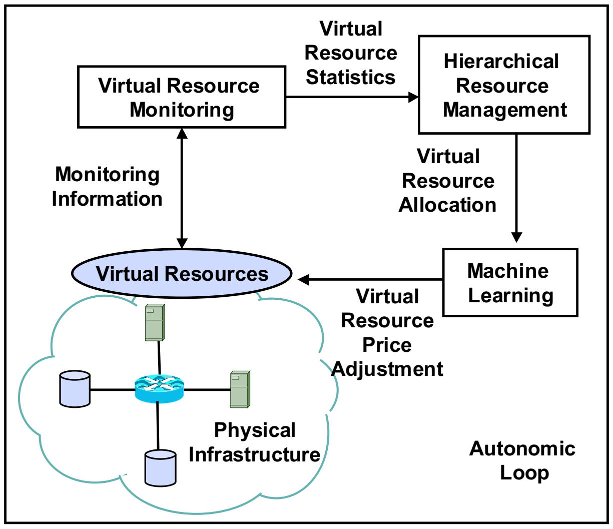
Virtual Resource Statistics
Hierarchical Resource Management
Virtual Resource Monitoring
Monitoring Information
Virtual Resource Allocation
Virtual Resources
Machine Learning
Virtual Resource Price Adjustment
Physical Infrastructure
Autonomic Loop

Fig. 2. Autonomic Hierarchical Resource Management Algorithm

The detailed hierarchical model of ARBs at a given SP is shown in Fig. 3. Initially, the SP reserves a fraction of the available capacity at its GARB for a given virtual resource. The CARBs will share this fraction of the capacity proportionally to the needs of the service that each CARB manages, while the remaining capacity is kept to remedy for any virtual resources shortages. Service requests arrive from customers at random times for the different services offered by this SP which use this virtual resource. Thus, a given amount of it is allocated to each activated service instance. These requests consist of two main tuples of interest for the management of a given virtual resource: <Amount, Duration>. The virtual resource amounts are allocated if available, and if not, a new request is triggered for additional virtual resources to be bought from other SPs.

The interaction between the different ARBs is shown in Fig. 4. At level 0, the GARB which is the global manager of a given SP, interacts with the GARBs of other SPs in a peer-to-peer (P2P) fashion. The SPs can therefore exchange virtual resources when needed. Initially, each SP has a maximum capacity array of the virtual resources $[MC(VR_1), \ldots, MC(VR_N)]$. At level 1, the CARB which manages a given service is allocated an array of virtual resource amounts $[A(VR_1), \ldots, A(VR_N)]$. This allocation is performed by the GARB to its children CARBs, i.e. the CARBs corresponding to the services offered by this SP.

Table 1. Notations for the market-based hierarchical resource management algorithm

R^r_{req}	Amount of Virtual Resource r requested by QoS customers
R^r_{BE}	Amount of Virtual Resource r offered to BE customers
$R^r_{miss}(ARB_i)$	Missing amount of Virtual Resource r at ARB i
$R^r_{thr}(ARB_i)$	Threshold amount of Virtual Resource r at ARB i
$RC(VR_r, ARB_i)$	Residual Capacity of Virtual Resource r at ARB i
$A_{upst}(VR_r, ARB_i)$	Total upstream amount of Virtual Resource r at ARB i
$A_{dnst}(VR_r, ARB_i)$	Total downstream amount of Virtual Resource r at ARB i
$A_{QoS}(VR_r, ARB_i)$	Total QoS amount of Virtual Resource r allocated at ARB i
$U^r(ARB_i)$	ARB i utility for Virtual Resource r
$\Delta U^r(ARB_i)$	ARB i differential utility for Virtual Resource r
p^r_{QoSi}	QoS price for Virtual Resource r at ARB i
p^r_{BEi}	BE price for Virtual Resource r at ARB i
$TRev(VR_r, ARB_i)$	Total revenue from Virtual Resource r at ARB i
$TCost(VR_r, ARB_i)$	Total cost for Virtual Resource r at ARB i
$\epsilon_1(t)$	Time varying scaling factor for utility function

The price of a virtual resource is inherited at the CARB from its parent GARB. We assume that a separate price is set by a SP for each virtual resource, one for BE requests and one for QoS requests. The price for a service composed of several virtual resources is a combination of these virtual resources' prices, according to service pricing approaches beyond the scope of this paper.

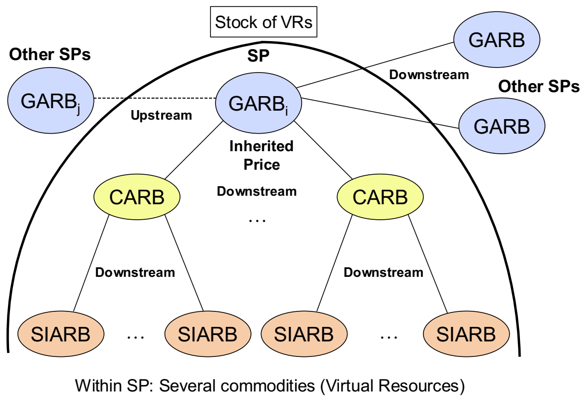

Fig. 3. Detailed Hierarchical Resource Management Model

The negotiation algorithm, shown in Algorithm 1, works as follows: For QoS customers, the SP needs to guarantee that the virtual resource amounts delivered to the customers are equal to those requested. For BE customers, the SP

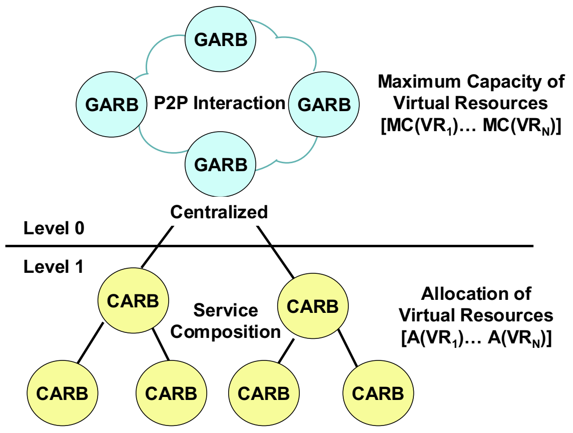
GARB
GARB
P2P Interaction
GARB
GARB
Maximum Capacity of Virtual Resources [MC(VR1)... MC(VRN)]
Centralized
Level 0
Level 1
CARB
Service Composition
CARB
Allocation of Virtual Resources [A(VR1)... A(VRN)]
CARB
CARB
CARB
CARB

Fig. 4. Autonomic Resource Brokers Interaction

does not guarantee delivery of any virtual resource amounts to the customers. An issue for the algorithm is to determine an appropriate utility function to maximize in order to determine the amount of virtual resources allocated to BE customers. The choice of the utility function was based upon the needs of an efficient management system. The higher the amount allocated, the greater the revenue for the SP. However, the amount allocated should not be too high, since it limits the number of additional customers that could be served in the future. One key requirement of the utility function is to be concave up. Due to these conflicting needs, we use two terms in the utility function: one relates to the revenue of the SP, and the other to the amount of resources allocated. Therefore, the chosen utility function for virtual resource r at $CARB_i$ is given by:

$$U^r(CARB_i) = \epsilon_1(t) \times log\left(1 - \frac{A_{dnst}(CARB_i) - A_{upst}(CARB_i)}{A_{dnst}(CARB_i) + RC(VR_r, CARB_i)}\right) + p^r_{BEi}(t) \times A_{dnst}(CARB_i) \quad (1)$$

The amount $\epsilon_1(t)$ is time-dependent and set to $p_{BEi}(t) \times (A_{dnst}(CARB_i) + RC(VR_r, CARB_i))$. In the rest of this paper, we are concerned with the additive utility, that is the utility added to the SP by allocating an amount R^r of virtual resource r to a customer, which is given by:

$$\Delta U^r(CARB_i) = \epsilon_1(t) \times log\left(1 - \frac{A_{dnst}(CARB_i) - A_{upst}(CARB_i) + R^r}{A_{dnst}(CARB_i) + RC(VR_r, CARB_i) + R^r}\right) - \epsilon_1(t) \times log\left(1 - \frac{A_{dnst}(CARB_i) - A_{upst}(CARB_i)}{A_{dnst}(CARB_i) + RC(VR_r, CARB_i)}\right) + p^r_{BEi}(t) \times R^r \quad (2)$$

The aforementioned utility function was chosen for the following reasons:

```
Hierarchical Resource Management Algorithm
Service Instance activation for Service $j$ by QoS Customer $i$ from Service Provider $k$
QoS Customers-Service Providers
Service Instance of Service $j$ activated;
Amount $R^r_{req}$ of virtual resource $r$ requested;
if $R^r_{req} < RC(VR_r, CARB_j)$ then
    Accept request for the amount $R^r_{req}$ of virtual resource $r$;
    Update CARB $j$ and SIARB $i$ accordingly;
else
    Find missing amount $R^r_{miss}(CARB_j) = R^r_{req}$ of virtual resource $r$ at
    CARB $j$;
    Trigger internal flow between CARB $j$ and GARB $k$ for missing
    amount $R^r_{miss}(CARB_j)$ of virtual resource $r$;
end
Service Instance activation for Service $j$ by BE Customer $i$ from Service Provider $k$
BE Customers-Service Providers
Service Instance of Service $j$ activated;
Find amount that maximizes the differential utility for virtual
resource $r$ at CARB $j$: $R^r_{BE} = argmax(\Delta U^r(CARB_j))$;
if $R^r_{BE} > R^r_{th}(CARB_j)$ then
    Accept Best-Effort request for amount $R^r_{BE}$ of virtual resource $r$;
else
    Find missing amount $R^r_{miss}(CARB_j) = R^r_{BE}$ of virtual resource $r$ at
    CARB $j$;
    Trigger internal flow between CARB $j$ and GARB $k$ for missing
    amount $R^r_{miss}(CARB_j)$ of virtual resource $r$;
end
CARB-GARB (Triggered)
Missing amount $R^r_{miss}(CARB_j)$ of virtual resource $r$ requested between
CARB $j$ and its parent GARB $k$
if $R^r_{miss}(CARB_j) < RC(VR_r, GARB_k)$ then
    Accept request for the missing amount $R^r_{miss}(CARB_j)$;
    Update CARB $j$ and GARB $k$ accordingly;
else
    Trigger external flow between GARB $k$ and other GARB $l$ for missing
    amount $R^r_{miss}(GARB_k) = R^r_{miss}(CARB_j)$ of virtual resource $r$;
end
Between GARBs (Triggered)
Missing amount $R^r_{miss}(GARB_k)$ of virtual resource $r$ requested between
GARB $k$ (or downstream GARB) and GARB $l$ (or upstream GARB)
if $R^r_{miss}(GARB_k) < RC(VR_r, GARB_l)$ then
    Accept request for the missing amount $R^r_{miss}(GARB_k)$;
    Update GARB $k$ and GARB $l$ accordingly;
else
    Reject triggered request, and original customer request;
end
```

Algorithm 1: Pseudo Code for Hierarchical Resource Management Algorithm

– The first derivative has a point of inflection where it changes signs.

$$\frac{dU^r(CARB_i)}{dR^r} = -\epsilon_1(t) \times \frac{1}{R^r + A_{dnst}(CARB_i) + RC(VR_r, CARB_i)} + p^r_{BEi}(t) = 0 \quad (3)$$

This point of inflection corresponds to the rate R^r_{BE} offered to the customer.

$$R^r_{BE} = \frac{\epsilon_1(t)}{p^r_{BEi}} - (A_{dnst}(CARB_i) + RC(VR_r, CARB_i)) \quad (4)$$

– The second derivative is always positive, so the proposed utility function is concave up, as required.

$$\frac{d^2U^r(CARB_i)}{dR^{r2}} = \frac{\epsilon_1(t)}{(R^r + A_{dnst}(CARB_i) + RC(VR_r, CARB_i))^2} \quad (5)$$

– In addition, the chosen utility function achieves our goal. An increase in the upstream amount is not desirable as this triggers external exchanges with other SPs, which is costly. When the residual capacity and the upstream amount both reach zero, the first utility term is equal to zero (its highest possible value), which means all the residual capacity is being used, but there is no need for additional resource amounts to be bought from other SPs.

5 Strategy Improvement using Learning

Reinforcement Learning (RL) represents a class of Machine Learning problems where an agent explores its environment, observes its current state s, and takes a subsequent action a, according to a decision policy $\pi : s \rightarrow a$. The RL model consists of a set of states S and a set of actions A. RL aims to find a control policy that will maximize the observed rewards over the lifetime of the agents, in our case the Global Autonomic Resource Brokers (GARBs) corresponding to the different service providers (SPs) in the network. To do so, a GARB will incrementally adjust its virtual resources' prices based on the feedback (or reinforcement) received from the environment. An optimal decision policy is to incur the highest accumulated reinforcement values.

Prior to defining the RL model, we need to clarify the goal that the network infrastructure aims to achieve. Ideally, no SP should monopolize the network, as customers should be tempted to buy from all SPs. However, SPs need to make as much profit as possible. By increasing their prices without boundary, SPs will be at a disadvantage if customers are looking for the cheapest service to buy and activate. In addition, customer demands should also be satisfied to the best of a SP's ability as long as no detrimental performance effects are observed.

In our approach, we used two different prices based on whether we are dealing with QoS or BE customers. The approach taken for each price adjustment method using RL could be different for the QoS and BE cases, but we assume a similar approach for both. Hence, in what follows, we will only show the QoS price adjustment approach. The BE price adjustment approach is similar, where BE prices are varied using BE virtual resource amounts instead of QoS virtual resource amounts to calculate the reinforcement value. Table 2 lists the variables used in the proposed RL method to improve hierarchical resource management.

Table 2. Variables used in Reinforcement Learning method

$Q^r(s^r, a)$	Q-value function for state s^r and action a
$P(a/s^r)$	Probability of taking action a when in state s^r
rl^r	Reinforcement value received by GARB
ψ^r	Positive constant to control exploration vs. exploitation
γ^r	Discounting factor
ζ^r	Learning rate
$\epsilon_2(t)$	Time varying scaling factor for reinforcement

RL operates on a virtual resource basis at a given GARB. In the discrete-time domain, RL models the interaction between a GARB and the environment as a Markov Decision Process. Suppose the GARB is in state s^r at time step (t), then the GARB performs action a and shifts to state $s^{r'}$ at the next time step $(t+1)$. In our case, the states are chosen as the ratio of the GARB's residual capacity for the given virtual resource, and the sum of the GARB's residual capacity and the delivered downstream QoS virtual resource amounts. The actions taken are variations of the QoS price $p^r_{QoSi}(t)$ charged by the upstream ARB to its downstream ARBs for QoS amounts of Virtual Resource r at time step (t), to steer the performance towards the desired goals.

In this paper, we will adopt the Q-learning algorithm to iteratively approximate the state-action value function, $Q^r(s^r, a)$, which represents the expected return when taking action a in state s^r and then following the current policy to the end. The action a in state s^r is taken with a probability $P(a/s^r)$, and the GARB receives a reinforcement value rl^r. The actions are picked according to their Q-values, following a Boltzmann distribution, as follows:

$$P(a/s^r) = \frac{\psi e^{Q^r(s^r,a)}}{\sum_{a'} \psi e^{Q^r(s^r,a')}} \tag{6}$$

The standard updating rules for Q-learning are given as follows:

$$Q^r(s^r(t+1), a) = (1-\zeta^r)\, Q^r\left(s^r(t), a\right) + \zeta^r\left(rl^r + \gamma^r max_{a'} Q^r\left(s^r(t), a'\right)\right) \tag{7}$$

At each GARB, the QoS price is dynamically adjusted over time to maximize its economic revenue and minimize its empirical loss due to the decrease of its residual capacity because of demands by downstream ARBs. Therefore, we choose the following reinforcement value for node i at time step $(t+1)$:

$$
\begin{aligned}
rl^r = \epsilon_2(t) \times \left(TRev\left(VR_r, GARB_i\right) - TCost\left(VR_r, GARB_i\right)\right) \\
+log\left(1 - \frac{A_{QoS}\left(VR_r, GARB_i\right)}{A_{QoS}\left(VR_r, GARB_i\right) + RC\left(VR_r, GARB_i\right)}\right)
\end{aligned}
\tag{8}
$$

The value $\epsilon_2(t)$ is chosen to be time-dependent in order to constantly adjust the reinforcement value as time elapses. In order to obtain similar orders of magnitude of the two terms of the reinforcement value, we set $\epsilon_2(t)$ to $500 \times p^r_{QoSi}(t)$. The reinforcement value is supposed to steer the SP towards the aforementioned goals, increasing its revenue, but also taking the residual capacity and the delivered rate into account. In order to ensure that the shortcomings which are usually encountered in RL models are avoided, and to guarantee convergence to global optima, the following conditions, satisfied in our RL model, are required:

1. Each state-action pair is visited an infinite (i.e. large) number of times
2. Learning rate is decreased with time
3. Rewards are incremented as the goal is approached

6 Simulation Results

The proposed approach is tested using extensive simulations. To emulate the desired environment of service providers (SPs), of physical and virtual resources, of services offered, of customers and the service instances they activate, we built a custom simulator using the Java programming language. We create customer entities connecting to the network infrastructure through physical resources to activate the service instances. Services are composed using other component services and several virtual resources. The candidate component services for the composition process are equally considered to avoid bias towards a given service.

The parameters used in the simulation were the same for all experiments. We generated several component services, as well as physical and virtual resources, which were owned by 10 SPs. For each virtual resource, random amounts were available in the infrastructure and distributed in the available physical resources. We allowed several customers to buy services from different SPs and to activate instances of such services. The service instances were activated according to a different Poisson process for each service bought, and were kept active for exponentially distributed service times which were different for each service instance.

Fig. 5 shows how the utility function chosen to service Best Effort (BE) customers works for a given SP. As can be seen in the figure, the rate offered varies depending on the virtual resource's residual capacity at this SP. When this residual capacity increases, the rate offered increases, and vice versa. This shows that the utility function is performing as desired, adapting the rate offered by this SP to BE customers according to its available capacity.

We now show the instantaneous variation of the QoS prices for 5 randomly chosen SPs for 2 cases where the Reinforcement Learning (RL) method is applied. In the Random Choice case (Fig. 6), the prices vary around their starting point.

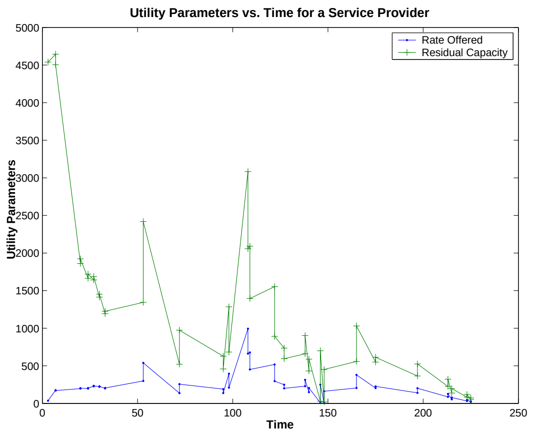

Fig. 5. Utility: Variation of Rate Offered with Residual Capacity

The interesting observation is that one SP's price tends to deviate from the others. This is due to the fact that in this experiment, the upstream SP when external flows are triggered is randomly chosen. There is therefore no reward for a SP offering a lower price for a given virtual resource, and it is not harmful in this case to deviate from the general consensus of keeping prices under control. The RL method is not totally suitable in this case. In the Least Cost case (Fig. 7), the prices vary around their starting point. The interesting observation is that no SP's price tends to deviate from the others. If it does, it is re-adjusted by the RL method. This is due to the fact that in this experiment, the upstream SP when external flows are triggered is chosen according to the least cost offered by other SPs for that virtual resource. There is therefore a reward for a SP offering a lower price for a given virtual resource, and it is harmful in this case to deviate from the general consensus of keeping prices under control. This is much more likely to be the case in real-world environments. In such environments, the proposed RL method for market-based hierarchical resource management performs as desired.

To further measure the performance of the proposed reinforcement learning approach, we propose a new metric which we refer to as the Virtual Resource Goodput (VRG). The VRG is calculated as the ratio of the total downstream rate of the SP over the revenue of that SP. Therefore, the goal of a SP is to bring the VRG down, so that less virtual resources are needed for more revenues. The value of his unit of virtual resource increases when the VRG decreases. Table 3 shows the average VRG for the three approaches: Fixed Price, Varying Price Random Choice, and Varying Price Least-Cost. We also compute the VRG for the QoS downstream rate only. As seen in the table, the average VRG is highest (worst performance) for the Fixed Price approach, followed by the Varying Price Random Choice approach, and the best performance is achieved by the Varying Price Least Cost approach. Also, note that the VRG of the QoS traffic is better than the VRG of the entire (QoS and BE) traffic, as the QoS traffic gives the SP more value per unit virtual resource.

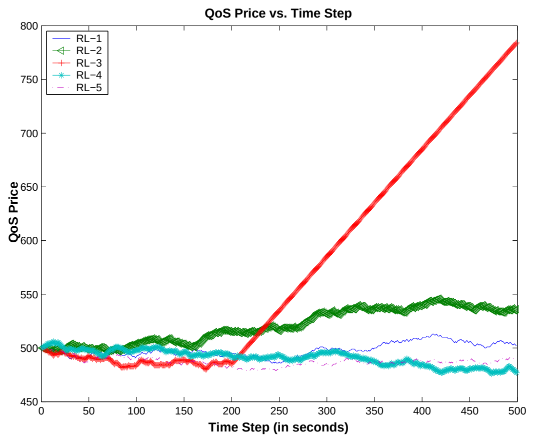

Fig. 6. QoS Price Variation for Random Choice Scenario

Table 3. Average Virtual Resource Goodput for the three pricing approaches

Traffic	Fixed	Varying Random Choice	Varying Least-Cost Choice
All Traffic	0.3083	0.2867	0.0657
QoS Traffic	0.2842	0.2441	0.0555

7 Conclusion

In this paper, we presented a market-based hierarchical resource management algorithm using machine learning to autonomically learn prices and adjust them as time evolves in order to improve the service providers' performance, keep the customers satisfied, and avoid monopolies by preventing service providers' price deviation. The paper proposed a negotiation algorithm using a carefully chosen utility function to serve BE customers, and providing QoS customers with the the rate requested. It also proposed a learning method to adjust virtual resource prices according to the environment. Results have shown that the utility function operates as expected, that the learning mechanism avoids price deviation and keeps prices under control, and that the virtual resource amount needed by

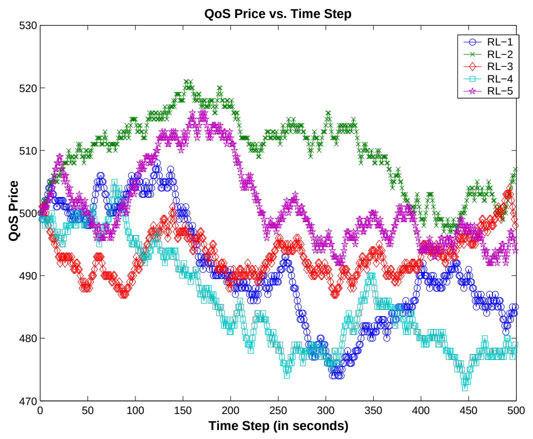

Fig. 7. QoS Price Variation for Least Cost Scenario

a given service provider for a unit of revenue decreases when learning is used. Future work will consider other similar hierarchical resource management algorithms, as well as variations to the utility functions and reinforcement learning methods. The paper has shown the potential of reinforcement learning when competing service providers wish to steer their operation towards a desired goal.

References

[1] Kephart, J. et. al. : The vision of autonomic computing. IEEE Computer Magazine. (2003) 41–50

[2] Farha, R. et. al. : Towards an Autonomic Service Architecture. Lecture Notes on Computer Science. (2005) 58–67

[3] Osborne, M. : An introduction to Game Theory. Oxford University Press. (2002)

[4] Alpaydin, E. : Introduction to Machine Learning. MIT Press. (2004)

[5] Kaelbling, L. et. al. : Reinforcement Learning A Survey. Journal of Artificial Intelligence Research. (1996) 237–285

[6] Wang, W. and Li, B. : Market-based self-optimization for autonomic service overlay networks, IEEE Journal on Selected Areas in Communications. (2005) 2320-2332

[7] Leon-Garcia, A. et. al. : Virtual Network Resource Management for Next-Generation Networks. IEEE Communications Magazine. (2003) 102–109

[8] Garfinkel, S. : Commodity Grid Computing with Amazon's S3 and EC2. Usenix. (2007)

[9] Minoli, D. : A networking approach to Grid Computing. Wiley. (2004)