J Internet Serv Appl (2012) 3:311–318
DOI 10.1007/s13174-012-0070-2

A distributed controller for a virtualized router

H. Mellah · O. Cherkaoui · Y. Lemieux

Received: 12 March 2011 / Accepted: 21 September 2012 / Published online: 16 October 2012

Abstract In this paper, a distributed controller for a virtualized router is proposed. This controller enables the dynamic and automatic resource allocation between the different virtual routers (called slices) running on top of the physical router. The controller is designed on a two-layer architecture. A slice controller (one for each slice) estimates the relationship between the past performances and resource allocations of the slice using a linear model, and then determines the requested allocation for the slice to meet its target performance. The physical router consists of a set of modular linecards. A resource controller (one for each linecard), collects the resource allocation requests from the different slices using the resources it controls and determines the allocations based on the available capacities of the resources. Resources are allocated to slices to guarantee their target performances if possible, or provide service differentiation if the total requests from all the slices exceeds the capacities of the shared resources. We have found that the convergence of the controller depends on different parameters (such as the number of slices and the parameters of the linear model) and therefore some tuning of these parameters is needed for the system to achieve the stability.

1 Introduction

Router virtualization is considered a promising solution for network providers and equipment vendors to better utilize their resources. It simplifies network design by allowing multiple virtual routers supporting multiple independent virtual networks running on the same shared physical substrate [1]. However, very little studies have dealt with issues related to router virtualization such as resource allocations. Most of the existing literature addressed virtualization in servers and data centers [2–11]. The objective of this paper is therefore to fill a part of that gap by proposing a distributed controller for dynamic resource allocation.

In a virtualized or sliced router, multiple virtual routers, called slices, co-exist on the same physical router platform and share its physical resources.

Due to the time-varying requirements and workloads of the different slices, resource management is particularly important aspect in a sliced router. Management functions should be performed dynamically to reflect any fluctuations in workloads or system conditions. They should also be performed automatically without the need for the administrator intervention.

In addition, resource allocation should be distributed, because the router resources used by the different slices may reside in different locations of the router line-cards. Furthermore, these linecards can be inserted or removed online which implies that if all the controller components are on a single linecard, removing this card results in the termination of the whole controller.

Resource allocation and management is executed periodically in specific control intervals. This automated management is performed by a two-layer controller. At the slice level, the slice controller records the past resource allocations and performance of the slice. Using the target performance of the current control interval, it estimates the new optimal resource allocations requested by the slice to meet its performance.

The requests from all the slice controllers are directed to the respective line-card and switch fabric controllers. The

H.Mellah (✉) · O.Cherkaoui
Computer Science Department, University of Quebec at Montreal (UQAM), Montreal, Canada
e-mail: hakim.mellah@ieee.org

Y. Lemieux
Ericsson Canada Inc., Montreal, QC, Canada

resource controllers at each line-card and switch fabric allocate resources to slices requesting them to achieve certain service level.

The remainder of the paper is organized as follow. In Sect. 2, a review of relevant related work is presented. The problem of automated resource allocation and the system overview are discussed in Sect. 3. In Sect. 4, the proposed controller is introduced and the different modules of this controller are detailed. Then in Sect. 5, simulations are conducted to evaluate the performance of the proposed model under different parameter values. Finally, Sect. 6 concludes the paper.

2 Related works

Control theory-based feedback loop has been recently applied for dynamic resource management in a virtualized environment. It guarantees system stability and can adapt to changes in workload and system conditions.

In Ref. [12], the authors applied control theory for designing a controllable computer system. They have derived a set of properties that have to be satisfied in order for the system to be controllable.

In Ref. [11], the authors proposed a multi-input multi-output (MIMO) controller for automated resource management in a virtualized data center. Their controller automatically adapts to dynamic workload changes to achieve service-level objectives for the different applications running on that virtualized infrastructure.

In another work [13], the authors have used control theory to design a dynamic system for provisioning of computing and storage resources in a cloud computing environment. Their system was modeled using an auto-regressive-moving-average with exogenous inputs (ARMAX) to represent the system behavior.

Controlling CPU and memory utilization in servers and data centers have been extensively addressed using control theory.

For example, the authors in Ref. [14] proposed to use a MIMO controller for CPU and memory utilization in a web server.

In Ref. [15], the authors proposed a proportional integral (PI) controller for admission control of client HTTP requests. This controller is implemented as a proxy, which operates by taking simple external measurements of the client response times.

In Ref. [16], the authors presented an adaptive controller that adjusts the CPU shares to individual tiers of multiple applications to meet a specified level of service differentiation. The controller parameters are automatically estimated using a recursive least-squares (RLS) method.

Our approach is similar to the one proposed in Refs. [11] and [13]. However, our controller is designed for a sliced router rather than a data center.

3 System model

Figure 1 shows a sliced aggregation router and Fig. 2 shows the logical architecture of its controller.

This router consists of three line-cards (LC1, LC2 and LC3) and a switch fabric (SF) shared by four slices ($S1$, $S2$, $S3$ and $S4$). The resources to be shared at each line-card are the network processor (NP) and the memory (M). For the

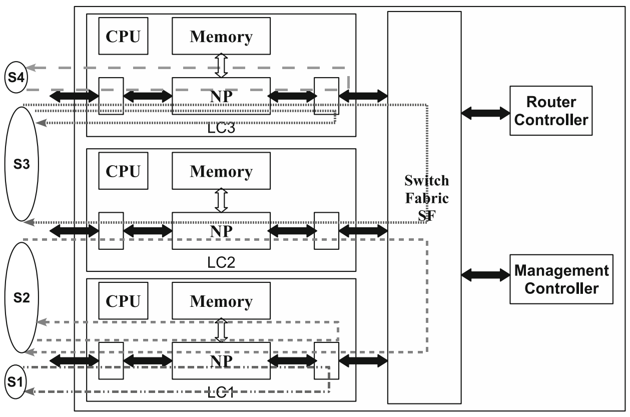


Fig. 1 An example of a sliced router

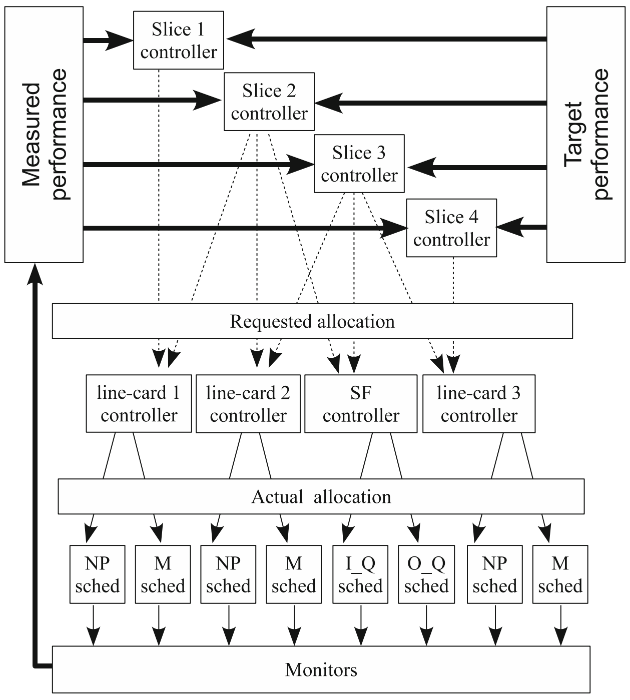


Fig. 2 The logical controller architecture

SF, the resources are the input queues (I_Q) and the output queues (O_Q).

Figure 2 gives the big picture of the logical working of the controller. Slice controllers receive the performances of the slices they control from the monitors, compare them with their desired (target) performance and then generate a request for resource allocation to the respective card controllers. The card controllers gather all the requests from the slices controllers and then compute the actual allocation for each slice. These actual allocations are used to schedule the resources for each slice. Once the resources are allocated to each slice, the monitors record the performances of the slices and then send them to the slice controllers.

Slice performance can be expressed through different performance metrics. We consider slice latency or response time as its performance metric. This is a good choice of the performance metric, since it reflects the processing power and the buffering capacity of the router. For each slice, we define the following parameters:

- LR_s is the total number of line-rates in slice s
- $xr^l_{s,r,c}$ is the fraction of resource type r in card c requested by line-rate l in slice s.
- $xr_{s,r,c}$ is the total allocation request of resource type r in card c by slice s, $0 \le xr_{s,r,c} \le 1$.

$$xr_{s,r,c} = \sum_{l \in \mathrm{LR}_s} xr^l_{s,r,c}$$

- $xa_{s,r,c}$ is the actual fraction of resource type r in card c allocated to slice s ($0 \le xa_{s,r,c} \le 1$).
- τ_s is the desired or target response time of slice s.
- t^i_{ps} is the ingress time of packet p. It represents the arrival time of the packet.
- t^e_{ps} is the egress time of packet p. It represents the departure time of the packet.
- $t_{ps} = t^e_{ps} - t^i_{ps}$ is the time spent by packet p in the router. It is the sum of all the processing and queuing times a packet has undergone inside the router.
- P_s is the total number of packets transmitted during a specific time interval (defined later as the control interval).
- M_s is the total number of packets in which $t_{ps} > \tau_s$. It represents the packets with a response time exceeding the slice target response time.
- $y_s = \frac{P_s - M_s}{P_s}$ is the measured performance of the slice. It represents the fraction of packets that met the slice target response time. This performance measure is at its maximum ($y_s = 1$) for a slice with all its packets meeting its desired response time.

For the controlled resources in line-cards that are dynamically allocated to slices, we consider network processor (NP) and memory (M). For the switch fabric, we consider the input queues (I_Q) and output queues (O_Q).

The controller has the objectives of assuring the performances of slices, providing service differentiation between the slices, automatically allocating resources, adapting to the variations in workloads and system conditions and scaling to many slices, line-cards and switch fabrics. Furthermore, it should converge (reaching a stable state) as quickly as possible in the case of any changes in workloads or system conditions. This convergence has to be also, as smooth as possible, avoiding any high degradation in the slices' performances.

4 The controller

As introduced in Sect. 1, the controller is designed based on a two-layer architecture. In this section, each control layer is discussed in detail. In designing our controller, we adopt a strategy similar to the one described in Refs. [11, 13]. Table 1

Table 1 The used notations

Symbol	Description
S	Set of hosted slices, e.g., $S = s_1, s_2, s_3, s_4$
LR_s	Set of line-rates in the slice s
C	Set of cards with controlled resources, e.g., $C = \{\mathrm{LC}_1, \mathrm{LC}_2, \mathrm{LC}_3, SF\}$
T_s	Tier of slice $s \in S$, e.g., $T_s = \{LC_1, LC_2, SF\}$
R	Set of all controlled resource types, e.g., $R = \{\mathrm{NP}, \mathrm{Memory}\}$
L_c	Set of slices sharing the resources of card c, e.g., $L_1 = \{s_1, s_2\}$
$xr^l_{s,r,c}$	Requested fraction of resource type r in card c to be allocated to line-rate l in slice s, $0 \le xr^l_{s,r,c}(k) \le 1$
$xr_{s,r,c}$	Requested fraction of resource type r in card c to be allocated to slice s (from all its line-rates), $0 \le xr_{s,r,c}(k) \le 1$
$xa_{s,r,c}$	Actual fraction of resource type r in card c allocated to slice s, $0 \le xa_{s,r,c}(k) \le 1$
τ_s	Target packet latency for slice s for a given time interval
ts	Measured average packet latency for slice s for a given time interval
y_s	Performance for slice s for a given time interval, where $y_s = \frac{\tau_s}{t_s}$
w_s	Priority weight for slice s
q	Stability factor in the slice controller

summarizes the different symbols and parameters used in the following discussion.

4.1 The slice controller

Each slice has a controller attached to it. This controller has two main functions. It first estimates a linear model to relate the resources allocated to the slice with its performance. This model uses both the past and present performance and allocations. Based on the estimated model, it predicts the resource allocation required to meet the performance target of the slice. Figure 3 shows the schematic diagram of the slice controller.

The model estimator learns and updates periodically a model for the dynamic relationship between the resource allocated to the slice and its performance using the following Eq. [12]:

$$y_s(k) = \mathbf{A}^T(k)\mathbf{Y}_s(k-1) + \mathbf{B}^{\mathrm{T}}(k)\mathbf{X}_s(k) \tag{1}$$

Where:

- $\mathbf{Y}_s(k-1) = [y_s(k-1), y_s(k-2)]^{\mathrm{T}}$ are the two past performances of slice s.
- $\mathbf{X}_s(k) = [\mathbf{xa}_s(k), \mathbf{xa}_s(k-1)]$ are the past and actual allocations for slice s.
- $\mathbf{xa}_s = [xa_{s,1}, xa_{s,2}, \ldots]^{\mathrm{T}}$ with $xa_{s,r}$ represents the allocation of resource r to slice s.
- $\mathbf{A} = [a_1, a_2]^{\mathrm{T}}$ are model parameters to capture the relationship between the past and the present performances of the slice.

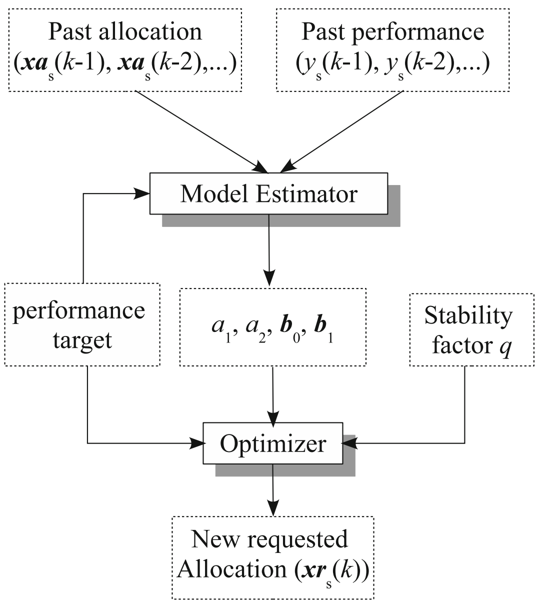

Past allocation
($\boldsymbol{xa}_s(k$-1), $\boldsymbol{xa}_s(k$-2),...)
Past performance
($y_s(k$-1), $y_s(k$-2),...)
Model Estimator
performance target
$a_1, a_2, \boldsymbol{b}_0, \boldsymbol{b}_1$
Stability factor q
Optimizer
New requested Allocation ($\boldsymbol{xr}_s(k)$)

Fig. 3 The slice controller internal architecture

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{b}_1 \end{bmatrix}$$

with $\mathbf{b} = [b^1, b^2, \ldots]^{\mathrm{T}}$ are the correlation vectors between the slice performance and its allocated resources.

Once the parameters of the linear model are computed, the optimizer predicts, based on the estimated model, the resource allocations ($\mathbf{xr}_s$) required for the slice s to meet its target performance ($y_s = 1$) in a stable way. To do so, the optimizer finds the value of $\mathbf{xr}_s$ that minimizes the following function [11]:

$$J_s = (y_s(k) - 1)^2 + q\|\mathbf{xr}_s(k) - \mathbf{xa}_s(k-1)\|^2 \tag{2}$$

The optimal resource allocation $\mathbf{xr}_s^*(k)$ that minimizes the cost function J_s can be derived from Eq. (1) and is given by:

$$\begin{aligned}\mathbf{xr}_s^*(k) = {} & (\mathbf{b}_0\mathbf{b}_0^T + qI)^{-1}((1 - a_1 y_s(k-1) - a_2 y_s(k-2) \\ & -\mathbf{b}_1^T \mathbf{xa}_s(k-1))\mathbf{b}_0 + q\mathbf{xa}_s(k-1))\end{aligned} \tag{3}$$

4.2 The resource controller

The resource controller (for line-card or SF) receives resource allocations requests from slice controllers and then determines the allocated resources to the slices according to the available resources. There are two possible scenarios.

- If the total requested resource allocation is less than the available capacity, the controller satisfies all the requests and then partitions the remaining extra capacity proportionally to the requests taking into consideration the slice priority weight w_s. Hence, the resource controller partitions resources as follows:

$$xa_{s,r,c} = xr_{s,r,c} + \Delta x_{r,c}\frac{w_s}{\sum_{i\in L_c} w_i} \tag{4}$$

  Where $\Delta x_{r,c}$ represents the extra resource capacity given by:

$$\Delta x_{r,c} = 1 - \sum_{i\in L_c} xr_{i,r,c} \tag{5}$$

- If the aggregated requested resources exceed the available capacity, the resource controller allocates resources to slices in such a way it locally minimizes the difference between the actual (from allocation) and the target performance values. This is achieved by minimizing a cost function that sums up the weighted square differences for all the slices sharing the resources, with the slice priority used as its weight. In this case, the actual allocations are found by solving the following problem.

$$\text{Minimize } J_c = \sum_{s \in L_c} w_s \left(\Delta x_{s,r,c}\right)^2 \tag{6}$$

s.t.

$$\sum_{s \in L_c} xa_{s,r,c} \le 1 \tag{7}$$

$$\Delta x_{s,r,c} \ge 0 \tag{8}$$

where $\Delta x_{s,r,c}$ represents the difference between the requested and the actual allocation of resources r for slice s given by:

$$\Delta x_{s,r,c} = xr_{s,r,c} - xa_{s,r,c} \tag{9}$$

Constraints (7) is the capacity constraint applied to actual resource allocation, whereas constraint (8) ensures that no slice exceeds its target which may cause other slices to be throttled. Service differentiation is achieved in this case by weighing the slice deficiencies by their priorities.

5 Performance evaluation

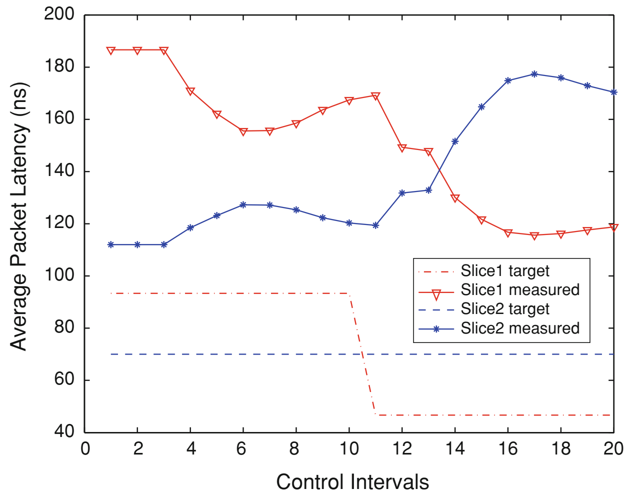
In this section, we present evaluation results of our controller and the effects of each parameter on its convergence. For this reason, we have used Matlab as the simulation tool. The results are based on an NP with a maximum throughput of 200 Mbps.

5.1 Service differentiation

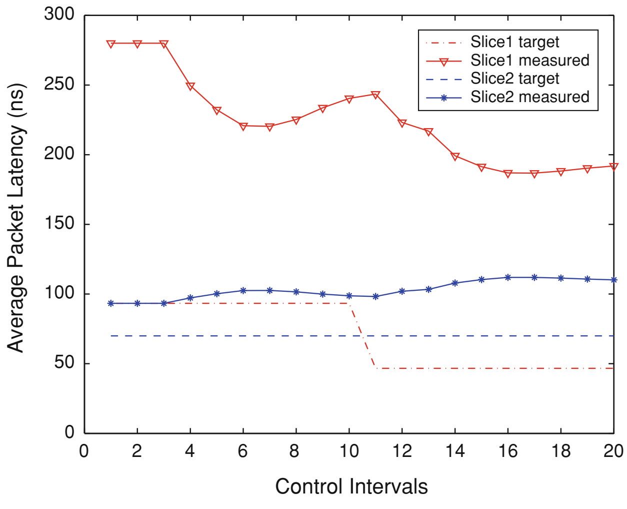
One of the objectives of the controller is to provide service differentiation between slices in the case of contention for resources. This situation is illustrated in Fig. 4. The figure shows the performances of two slices (in terms of their average packet latency) contending to share one resource in two different cases. In both cases, performance target of slice2 is kept constant for the entire control intervals, however, the target performances of slice1 is halved after half the control intervals. The only changes between the two cases is that in the first case, the two slices have equal priorities and in the second case, slice2 has higher priority than slice1.

As shown in Fig. 4, the performance (latency) of slice2 is increased after the target of slice1 has been decreased. This increase in the packet latency of slice2 is caused by the decrease in packet latency of slice1. This is due to the equal priorities of the slices, therefore, the one with higher target (request) receives the higher latency.

In Fig. 4, slice2 has higher priority than slice1, so it receives better performances than slice2 even in the case when the target for slice2 is higher than its target. So, the slices with high priority always receive better performance than the slices with low priority. This shows the service differentiation provided by the controller.



(a) No priorities



(b) With priorities

Fig. 4 Service differentiation (average packet latency)

5.2 The number of slices

The controller convergence depends on the number of slices running on the system. When there is a high number of slices competing for the same resource, more control intervals are required for the system to converge, since the slice controllers are working independently from each other and independent from the card controllers, they always try to get the required resource allocations to reach their targets. This allocation is determined by the card controller, which has a global view of the resource capacity as well as the slices requests. The

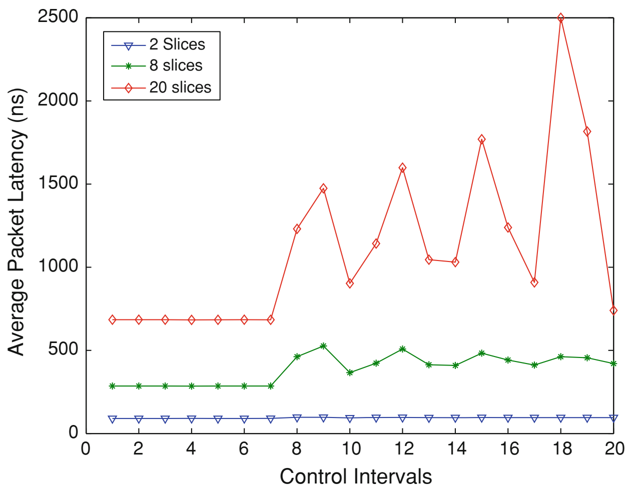


Fig. 5 The effect of the number of slices on the controller convergence

card controller then applies equity between slices and service differentiation when allocating the resources. Therefore, the number of control intervals needed for achieving a steady-state is increasing with the increase in the number of slices.

Figure 5 illustrates the impact of the number of slices on the stability of the controller. The figure shows the performance of one slice in the case of 2, 8 and 20 slices in the system (having equal priorities). When there are only two slices contending for the resources, the controller adjusts the resource allocation in a smooth way. When the number of slices increases, the convergence of the controller starts to become an issue causing the performances of the slices to oscillate. The more the slices, the worst the convergence of the controller. Furthermore, this oscillation can bring the system into an unstable state. This instability problem can be solved by tuning some controller parameters, such as increasing the value of the stability parameter in the slice controllers.

5.3 The stability parameter q

As discussed in the previous section, the increase number of slices in the system may cause convergence problem. This stability issue may be overcome by manipulating the value of the stability parameters of the slice controllers. However, taking a high value of this parameter makes the controller lazy and it does not react adequately to changes in slices workloads.

Figure 6 illustrates the effect of the stability factor on the convergence of the system. The figure shows the case of 20 slices sharing the same resources with different values of q. In case where the value of the parameter is small ($q = 0.6$), the controller is actively tracking any changes in the workloads of the slices and tries to guarantee the desired target for each slice. This leads to oscillations in slice performances.

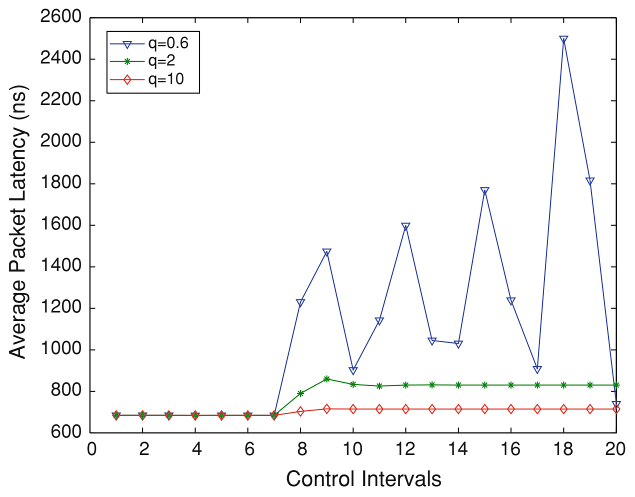


Fig. 6 The effect of the value of q on the controller convergence

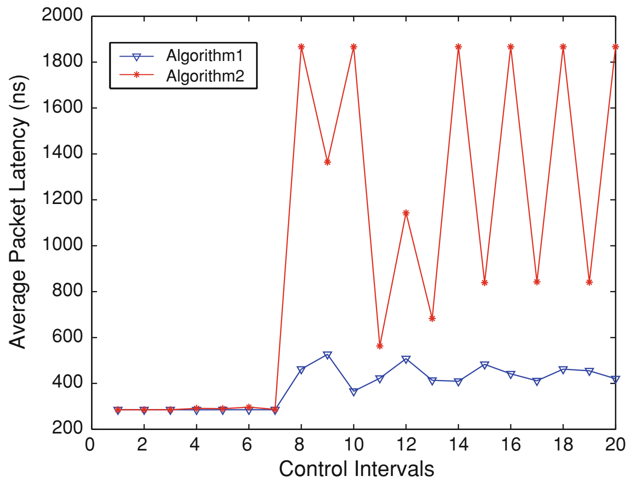


Fig. 7 The effect of estimation of the slice controller parameters

If the value of the parameter is high (e.g., $q = 10$), the controllers react less aggressively to changes in workloads and the stability of the system is achieved. However, this stability comes from the fact that the system is passive to changes in workloads and may not allocate resources in an optimal way between the slices.

5.4 The system estimator

Another important factor impacting the convergence of the system is the way to estimate the linear model parameters in Eq. (1).

Figure 7 illustrates the impact of the estimation algorithm for the slice controller parameters updates. We have used two different algorithms to estimate the parameters of the linear model. In the first algorithm (algorithm1), we have used an approximation for the recursive least-squares (RLS) algorithm and in the second case (algorithm2), we have used a built-in function in Matlab called recursive ARMAX for the estimation of the parameters of a recursive ARMA model. It is clear from the figure that the accurate estimation of the parameters is very important in the stability of the system. In this figure, when we have used a built-in function, which is not well adapted to our model, the system becomes unstable. However, the the case of the a well-adapted algorithm (algorithm1), the model makes good estimations of the system parameters which results in a more stable behavior in the system performance.

6 Conclusion

In this paper, we have proposed a distributed controller based on a feedback-loop control system for resource allocation in a sliced router. The controller is a two-layer architecture composed of a set of slice controllers and a set of card controllers. Slice controllers estimate the relationship between the past and current allocations and performances. This estimation is very important for the computation of the optimal request for the next control interval. If the system parameters are wrongly estimated, the system may diverge from the target performance. In our system, the model parameters have been estimated using a RLS adaptive filter, which exhibits extremely fast convergence.

The controller convergence depends also on the number of slices in the system. Naturally, a high number of slices competing for the same resource need more control intervals to converge, since the slice controllers are working independently from each other and independent from the card controllers, they always try to get the required resource allocations to reach their targets. This allocation is determined by the card controller, which has a global view of the resource capacity as well as the slices requesting it. The card controller then applies equity between slices and service differentiation when allocating the resources. Therefore, the number of control intervals needed for achieving a steady-state is increasing with the increase in the number of slices.

Another important parameter for the stability of the system is the length of the control interval. When the control interval is small, any change in workload can be captured by the controller. However, if the control interval is very short, a large number of updates are needed and a huge amount of messages are exchanged between the different controllers, which results in a system overhead. In addition to that, very short control intervals may cause some of the system parameters not to be available at the next control period. On the other hand, if the control interval is very long, the system cannot cope with the changes in workload and may result in performance degradation of the system.

The stability of the controller depends also on the stability parameter of the slice controller. With small values of this parameter, the controller reacts aggressively to any changes in workload. This results in performance oscillations. However, for high values of q, the controller becomes very slow in detecting and adjusting resource allocations to meet the target performances.

The system parameters and its stability depends also on the type of applications and resources. When the traffic pattern of the slice changes very fast, the system parameters should be chosen to react quickly to those change and then adjust resource allocations accordingly. This implies a small control interval and a small value of the stability parameter.

References

1. Wang Y, van der Merwe J, Rexford J, (2007) VROOM: Virtual ROuters On the Move. In: Proceedings of ACM SIGCOMM workshop on hot topics in networking, pp 1–7
2. Chowdhury NM, Boutaba R (2010) A survey of network virtualization. Comput Netw 54(5):862–876
3. Almeida J, Almeida V, Ardagna D (2010) Joint Admission control and resource allocation in virtualized servers. J Parallel Distrib Comput 70(4):344–362
4. Ardagna D, Panicucci B, Trubian M, Zhang L (2010) Energy-aware autonomic resource allocation in multi-tier virtualized environments. In: IEEE transactions on services computing, 2010
5. Urgaonkar R, Kozat UC, Igarashi K, Neely MJ (2010) Dynamic resource allocation and power management in virtualized data centers. In: IEEE network operations and management symposium (NOMS), 2010, pp 479–486
6. Chowdhury NMMK, Boutaba R (2009) Network virtualization: state of the art and research challenges. IEEE Commun Mag 47(7):20–26
7. Ardagna D, Trubian M, Zhang L (2007) SLA based resource allocation policies in autonomic environments. J Parallel Distrib Comput 67(3):259–270
8. Kusic D, Kephart JO, Hanson JE, Kandasamy N, Jiang G (2009) Power and performance management of virtualized computing environments via lookahead control. Cluster Comput 12(1):1–15
9. Wang X, Du Z, Chen Y, Li S (2008) Virtualization-based autonomic resource management for multi-tier web applications in shared data center. J Syst Softw 81(9):1591–1608
10. Bennani MN, Menasce DA (2005) Resource allocation for autonomic data centers using analytic performance models. In: Proceedings of second international conference on autonomic computing, ICAC 2005, pp 229–240
11. Padala P, Hou KY, Shin KG, Zhu X, Uysal M, Wang Z, Singhal S, Merchant A (2009) Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European conference on computer systems, 2009, pp 13–26
12. Karamanolis C, Karlsson M, Zhu X (2005) Designing controllable computer systems. In: Proceedings of the 10th conference on hot topics in operating systems, vol 10
13. Zhu Q, Agrawal G (2010) Resource provisioning with budget constraints for adaptive applications in cloud environments. In: Proceedings of the 19th ACM international symposium on high performance, distributed computing, 2010, pp 304–307
14. Gandhi N, Tilbury DM, Diao Y, Hellerstein J, Parekh S (2002) MIMO control of an apache web server: modeling and controller

Ddesign. In: Proceedings of the 2002 American control conference, 2002, vol 6, pp 4922–4927

15. Kamra A, Misra V, Nahum EM (2004) Yaksha: a self-tuning controller for managing the performance of 3-tiered web sites. In: Twelfth IEEE international workshop on quality of service, IWQOS 2004, pp 47–56

16. Liu X, Zhu X, Padala P, Wang Z, Singhal S (2007) Optimal multi-variate control for differentiated services on a shared hosting platform. In: 46th IEEE conference on decision and control 2008, pp 3792–3799