Load balancing vs. distributed rate limiting: an unifying framework for cloud control

Rade Stanojević, Robert Shorten
Hamilton Institute, NUIM, Ireland

***Abstract*— With the expansion of cloud-based services, the question as to how to control usage of such large distributed systems has become increasingly important. *Load balancing* (LB), and recently proposed *distributed rate limiting* (DRL) have been used independently to reduce costs and to fairly allocate distributed resources. In this paper we propose a new mechanism for cloud control that unifies the use of LB and DRL: LB is used to minimize the associated costs and DRL makes sure that the resource allocation is fair. From an analytical standpoint, modelling the dynamics of DRL in dynamic workloads (resulting from LB cost-minimization scheme) is a challenging problem. Our theoretical analysis yields a condition that ensures convergence to the desired working regime. Analytical results are then validated empirically through several illustrative simulations. The closed-form nature of our result also allows simple design rules which, together with extremely low computational and communication overhead, makes the presented algorithm practical and easy to deploy.**

***Index Terms*— Rate limiting, Load balancing, CDN, Cloud control.**

I. Introduction

In the early days of the Internet, services were centric, meaning that a user connects to the specific location that provides the service. Recently, we have seen a trend of moving from a centric model of providing services to a so called cloud-based model in which a user obtains a service from a massive network of "cloud servers". Nowadays, many internet services are structured in a "cloud" around a large number of servers that are distributed worldwide to decrease the costs and to improve content availability, robustness to faults, end-to-end delays, and data transmission rates [8]. Examples include most of Yahoo! and Google services, Amazon's Simple Storage Service (S3) and Elastic Compute Cloud (EC2) as well as Akamai's Content-Delivery Network (CDN). Some other applications, such as Google Docs or Microsoft Groove Office, have integrated software-as-a-service paradigm and allow desktop users to utilize cloud-based services in hosted environments.

The ability to control cloud-based service usage is critical for several important functions of a cloud-based service provider (CBSP):

(1) *The pricing* of service by most of the existing CBSPs is usage-based [1], [19]. Namely, services are charged at a rate that is an increasing function of the total resources used. However, in the history of communications, pricing of various services (eg. ordinary mail, the telegraph, the telephone, and the Internet) followed similar pattern: it started with usage-based pricing and converged to some form of flat-fee pricing. Moreover, enterprizes tend to prefer a fixed cost of an IT service rather than unlimited/unpredictable usage-based cost, see [9] and [18].

(2) *Provisioning* of high quality services depends on the nature of the service demand pattern. The ability to regulate the usage of individual service allows CBSPs to design networks with predictable performance bounds.

(3) *Fault tolerance* of large-scale distributed services is an important performance objective that is enhanced by resource control by means of fast fault discovery and quick response to these faults.

The standard approach for enforcing service usage of a particular subscriber[1] in the cloud would be enforcing some form of *load-balancing* (LB) [8]. More recently, a new mechanism, called *distributed rate limiting* (DRL) is proposed to tackle the same problem assuming that virtually infinite capacity is available at each server for each customer. In the following paragraph we briefly discuss the assumption behind LB and DRL.

A. Motivation: LB versus DRL

Load balancing is a general mechanism for allocating jobs to different servers such that load is "equalized". More formally, the set of jobs $\mathcal{D}$ needs to be processed with N available servers with (fixed) capacities $\mu_1, \dots, \mu_N$. The problem of load-balancing can be formulated as finding the partition of $\mathcal{D} = \cup_{i=1}^{N} \mathcal{D}_i$, so that jobs from $\mathcal{D}_i$ are served by server i and the performance at all servers is (approximately) equalized.

Load balancing algorithms, in general, assume a uniform price of allocating job j to server i (from now on we call this *uniform cost assumption*); see for example [20]. However it is often desirable (for example in the context of content delivery, VoIP or online interactive games) to connect the users to the "closest" server that can provide service, as this reduces cost to the service provider, and can potentially improve the quality of experience to the end users. As a result of this, many cloud-based service providers chose to serve a particular user from the nearest possible server[2]. Such job-server allocation might have highly nonuniform per-server demand. The distributed rate limiting paradigm is proposed in [19] to tackle this nonuniform per-server demand in a manner that equalize the performance among the servers while at the same time cap the

[1]The subscriber is either an enterprize, bank, corporation, or any other set of users paying for service under one account.

[2]This is typically implemented through the mechanism called DNS redirection, see [13].

aggregate bandwidth usage at a constant level. More formally, in DRL the demand set $\mathcal{D}$ is partitioned in N subsets $\mathcal{D} = \cup_{i=1}^{N} \mathcal{D}_i$ (based on the cheapest neighbor rule). Then the DRL problem translates into computing capacities $(\mu_1, \mu_2, \ldots, \mu_N)$ of servers such that performance at all servers is equalized and aggregate capacity constraint is satisfied:

$$\sum_{i=1}^{N} \mu_i = \mu = const.$$

The basic assumption behind the distributed rate limiting model is the *infinite capacity assumption* in which each cloud based provider is assumed to have virtually infinite bandwidth compared to the existing demands. This over-provisioning assumption might be valid in some monopolized markets. However, its validity is questionable in markets in which service providers compete heavily for each subscriber.

In this paper we consider the problem of cloud control in a model that does not assume infinite capacity nor uniform cost. The resulting framework unifies load balancing and distributed rate limiting paradigms. A practical algorithm is proposed to solve the optimization problem arising from the framework. The optimization problem strives to distribute the load at all servers such that associated costs (of using certain job-server allocation as well as cost of having certain demand level at a server) is minimized.

Comment 1: An early DRL scheme has been proposed in [11]. More recent work on DRL and LB includes: work on distributed threshold crossing algorithms [24]; overlay resource control based on two-random choice [15] load balancing [17]; distributed emulation of single best-effort and processor sharing queues [22].

B. Problem formulation

Suppose that we have M jobs with demands $d_1, \ldots, d_M$ and N processors with aggregate service rate μ, with the load

$$\lambda = \frac{\sum_{j=1}^{M} d_j}{\mu} < 1.$$

Denote by C_{ij} the cost of using one capacity-unit (say bit-per-second) of server i by job j. Each user (job) j can allocate its demand d_j to any of the servers, and by $x_j^{(i)} \geq 0$ we will denote the fraction (see Section IV for more details) of d_j served by server i. Thus we have:

$$\sum_{i=1}^{N} x_j^{(i)} = d_j. \tag{1}$$

Denote

$$y_i = \sum_{j=1}^{M} x_j^{(i)}$$

as the total demand at server i and by $D_i(y_i)$ the cost of using server i when the demand is y_i.

Each server has the capacity μ_i adaptable in a distributed manner such that

$$\sum_{i=1}^{N} \mu_i = \mu. \tag{2}$$

The performance at server i is measured through the performance indicator q_i (q_i can stand for any performance metric such as “available bandwidth”, drop-rate, mean-response-time, etc.) that is a function of the demand y_i and capacity μ_i:

$$q_i = f(y_i, \mu_i) = f_i(\mu_i). \tag{3}$$

The problem is then to find the matrix of demand allocations $\mathbf{x} = \{x_j^{(i)}\}$, and vector of server capacities $\mu = \{\mu_i\}$ for which the performance at each server is (approximately) equal and that minimizes the aggregate cost:

$$V(\mathbf{x}) = \sum_{j=1}^{M} \sum_{i=1}^{N} C_{ij} x_j^{(i)} + \sum_{i=1}^{N} D_i(y_i). \tag{4}$$

We allow each server to exchange information with some small set of neighbors in the (undirected, connected) communication graph G. Further, we make the following technical assumption that will allow us to use convex programming:

Assumption 1: The cost functions $D_i(\cdot)$ are convex.

Under above Assumption it is easy to see that $V(\mathbf{x})$ is a convex function of parameter $\mathbf{x}$ as well.

C. Our contributions

The main concern of this paper is the development of the framework for usage control in cloud-base services. Briefly, the main contributions of our work are following:

- We propose the framework for usage control of cloud-based services that takes into account the appropriate cost of job-server allocations and generalizes the well-known concepts of load balancing and distributed rate limiting.
- An algorithm for solving the optimization problem of interest is proposed. It relies on standard subgradient method and a novel DRL scheme that adapts server capacities in the dynamic workloads (resulting from the subgradient algorithm for demand allocations) so that performance at all servers is equalized.
- An analysis of the dynamical system modelling the per-server performance evolution is performed for a particular choice of performance metric. A sufficient condition is derived which guarantees that the algorithm runs system to the desired state.
- Several illustrative simulations are presented to evaluate the behavior of the algorithm and support our analytical findings.

```
DRL-UpdateCapacities()
    Once every $\Delta$ units of time do
        for $i = 1 : N$
            $\mu_i \leftarrow \mu_i + \eta \sum_{(i,j)\in E}(q_i - q_j)$
        endfor
    enddo

InitializeCapacities()
    for $i = 1 : N$
        $\mu_i \leftarrow \frac{\mu}{N}$
    endfor

UpdateDemands()
    $g(k) =$ Subgradient of $V$ in $\mathbf{x}(k)$
    $\mathbf{x}(k+1) = \mathbf{x}(k) - \frac{1}{\sqrt{k}} g(k)$
```

Fig. 1. Pseudo-code of CARTON

II. CARTON: Unifying Load Balancing and Distributed Rate Limiting

CARTON is the scheme that solves the optimization problem described in Section I-B. It has two steps. First is the subgradient method [5] that allocates the workloads $\mathbf{x}$ such that the cost function V is minimized. And second, a DRL algorithm that allocates the server capacities $(\mu_1, \ldots, \mu_N)$ in a manner that ensures that performance levels (measured through performance indicators q_i) at all the servers are equal. The pseudocode of the algorithm is given in Figure 1.

Subgradient methods are well established techniques for solving the convex optimization problems; we refer the interested readers to [5], [16]. The particular choice of step size used in the CARTON: $\alpha(k) = \frac{1}{\sqrt{k}}$ ensures convergence to the optimal point.

The DRL step updates the server capacities in discrete time steps by using the rule: $\mu_i \leftarrow \mu_i + \eta \sum_{(i,j)\in E}(q_i - q_j)$. The rationale for this update step is the following. q_i - the performance indicator of the quality of service is a decreasing function of μ_i. If the q_i at limiter i is higher than performance indicator q_j at some neighbor j of i (in G), then this indicates that some extra capacity should be allocated to limiter i which should be compensated by reducing the capacity of limiter j. Giving more capacity to limiters with high performance indicators affects improving the performance at those limiters. The parameter $\eta > 0$ determines responsiveness and stability properties. While the basic algorithm makes sense intuitively, many questions need to be answered before it can be deployed. Paramount among these concerns under which conditions does the algorithm CARTON converge to the desired (unique) state.

While many performance metrics can be defined (for example: drop rate, mean response time, etc.), to ease the exposition, in the rest of the paper we will use the simple "available-bandwidth" performance indicator:

$$q_i = f(y_i, \mu_i) = y_i - \mu_i. \tag{5}$$

For the analysis under general performance indicator functions, see the technical report [23]. The following theorem gives a sufficient condition on η that ensures that performance indicators eventually become equal.

Theorem 1: Let l_i be the degree of limiter i in the communication graph G. Then if η satisfies:

$$0 < \eta < \min_{1\le i\le N} \frac{1}{l_i}, \tag{6}$$

then for all i, j

$$\lim_{k\to\infty} q_i(k) - q_j(k) = 0.$$

Proof: We use the notation introduced in Section I-B. The performance indicator $q_i(k)$ is given by

$$q_i(k) = y_i(k) - \mu_i(k) = \sum_{j=1}^{M} x_j^{(i)}(k) - \mu_i(k).$$

Since $\mathbf{x}(k)$ converges to $\mathbf{x}^*$, the sequence $y_i(k)$ also converges to an appropriate value y_i^*. Further, the evolution of the vector $\mu = (\mu_1, \ldots, \mu_N)$ can be written as

$$\mu_i(k+1) = \mu_i(k) + \eta \sum_{(i,j)\in E} (q_i(k) - q_j(k))$$

or

$$q_i(k+1) = q_i(k) - \eta \sum_{(i,j)\in E} (q_i(k) - q_j(k)) + y_i(k+1) - y_i(k).$$

Therefore, the evolution of the state-vector $\mathbf{q}(k) = (q_1(k), \ldots, q_N(k))$ can be written as:

$$\mathbf{q}(k+1) = B\mathbf{q}(k) + \mathbf{u}(k), \tag{7}$$

where

$$\mathbf{u}(k) = \mathbf{y}(k+1) - \mathbf{y}(k) \to 0,$$

and where the matrix B is given by $B =$

$$\begin{bmatrix} 1 - l_1\eta & \eta e_{1,2} & \cdots & \eta e_{1,N} \\ \eta e_{2,1} & 1 - l_2\eta & \cdots & \eta e_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \eta e_{N,1} & \cdots & \cdots & 1 - l_N\eta \end{bmatrix}$$

with $e_{i,j}$ being the elements of the adjacency matrix of G, ie. if $(i,j) \in E$, then $e_{i,j} = 1$ otherwise $e_{i,j} = 0$. Then, for η satisfying the condition (6) the matrix B is a primitive nonnegative matrix [2]. In other words there is a positive integer s such that all elements of matrix B^s are strictly positive. Denote by δ the smallest element of the matrix B^s and by

$$\beta(k) = \max_{1\le i\le N} q_i(k) - \min_{1\le i\le N} q_i(k).$$

Now we prove that for any $\epsilon > 0$ there is $K_1 > 0$ such that for all $k > K_1$: $\beta(k) < \epsilon$.

Since $\lim_{k\to\infty} \mathbf{u}(k) = 0$, there is a K_0 such that for all $k > K_0$,

$$\max_{1\le i\le N} |u_i(k)| < \epsilon_1 := \frac{\epsilon\delta}{2s(1+\delta)}.$$

Then since B is a row-stochastic matrix, $\max_{1\le i\le N} q_i(k) \ge \max_{1\le i\le N} q_i(k+1)$ and $\min_{1\le i\le N} q_i(k) \le \min_{1\le i\le N} q_i(k+1)$ which (together with bound $\max_{1\le i\le N} |u_i(k)| < \epsilon_1$) implies that for all $k > K_0$:

$$\beta(k+1) \le \beta(k) + 2\epsilon_1. \tag{8}$$

On the other hand, since B is a stochastic matrix for all positive integers r, and any $k > K_0$, absolute values of the maximum and minimum of components of vector $B^r\mathbf{u}(k)$ are not greater than ϵ_1. If we denote by $\gamma(k) = B^s\mathbf{q}(k)$, then:

$$\max_{1\le i\le N} \gamma_i(k) \le (1-\delta)\max_{1\le i\le N} q_i(k) + \delta \min_{1\le i\le N} q_i(k)$$

and

$$\min_{1\le i\le N} \gamma_i(k) \ge (1-\delta)\min_{1\le i\le N} q_i(k) + \delta \max_{1\le i\le N} q_i(k)$$

therefore

$$\max_{1\le i\le N} \gamma_i(k) - \min_{1\le i\le N} \gamma_i(k) \le (1-2\delta)\beta(k).$$

Now, from (7)

$$\mathbf{q}(k+s) = B^s\mathbf{q}(k) + \sum_{r=0}^{s-1} B^r\mathbf{u}(k+s-1-r).$$

Then from above inequalities we conclude that for all $k > K_0$:

$$\beta(k+s) \le (1-2\delta)\beta(k) + 2s\epsilon_1. \tag{9}$$

Note that from (9): if $\beta(k) \ge 2s\epsilon_1/\delta$ then

$$\beta(k+s) \le (1-\delta)\beta(k),$$

implying that there is $K_1 > K_0$ such that

$$\beta(K_1) < \frac{2s\epsilon_1}{\delta}. \tag{10}$$

Now we prove that for all $k > K_1$, $\beta(k) \le \frac{2s\epsilon_1}{\delta} + 2s\epsilon_1 = \epsilon$. Indeed, from (8) and (10) we have that for $0 \le r \le s$:

$$\beta(K_1+r) \le \beta(K_1) + 2r\epsilon_1 \le \frac{2s\epsilon_1}{\delta} + 2s\epsilon_1.$$

Suppose now that there is $k > K_1$ for which $\beta(k) > \epsilon$, and denote by m the smallest index k for which $\beta(k) > \epsilon$. Note that from above inequality we have that $m > K_1 + s$. Then:

$$\beta(m-s) \ge \frac{\beta(m) - 2s\epsilon_1}{1-2\delta} > \frac{\frac{2s\epsilon_1}{\delta}}{1-2\delta} > 2s\epsilon_1\frac{1}{\delta(1-\delta)} >$$
$$> 2s\epsilon_1\frac{1-\delta^2}{\delta(1-\delta)} = 2s\epsilon_1\frac{1+\delta}{\delta} = \epsilon,$$

thus contradicting the assumption that m is the smallest index greater than K_1 for which $\beta(m) > \epsilon$. This contradiction completes the proof. ∎

III. Evaluation

A. Static demands

The basic setup is following. There are $M = 200$ jobs serviced by $N = 10$ servers. Each job and each server is located in a random point uniformly drawn in the unit square. The price, C_{ij} of using unit-capacity of resource i by job j is the euclidian distance between the points corresponding to job j and server i. The demand of each job is $d_j = 1$ and the aggregate capacity is such that load $\lambda = 0.9$. We run CARTON in two cases based on the cost functions D_i. The subgradient algorithm and DRL capacity allocation algorithm are performed independently: the first one minimizing the cost $V(\cdot)$ and the second enforcing the uniform performance among servers. In order to measure how "equal" performance defined by (5) (ie. the components of the vector $\mathbf{q}(k)$) at servers are, we use the *Jain's fairness index* (JFI)[12] defined as:

$$JFI(\mathbf{q}(k)) = \frac{\left(\sum_{i=1}^N q_i(k)\right)^2}{N\sum_{i=1}^N q_i^2(k)}. \tag{11}$$

JFI is a quantity that lies between 0 and 1, and the closer JFI is to 1 the "more uniform" elements of the vector are. The DRL topology graph is obtained as the minimum-cost spanning tree (MCST) and the parameter η that determines the responsiveness of the algorithm is chosen at the value that guarantees the stability (6).

Case 1. $D_i \equiv 0$. We expect that in this case all jobs connect to the "cheapest" (nearest) resource and that the DRL algorithm adapts the capacities to the values that ensure uniform per-server performance. The dynamics of the cost function $V(\mathbf{x}(\text{k}))$ as well as $JFI(\mathbf{q}(k))$ is depicted in Figure 2. As we can see, both quantities converge quickly to the

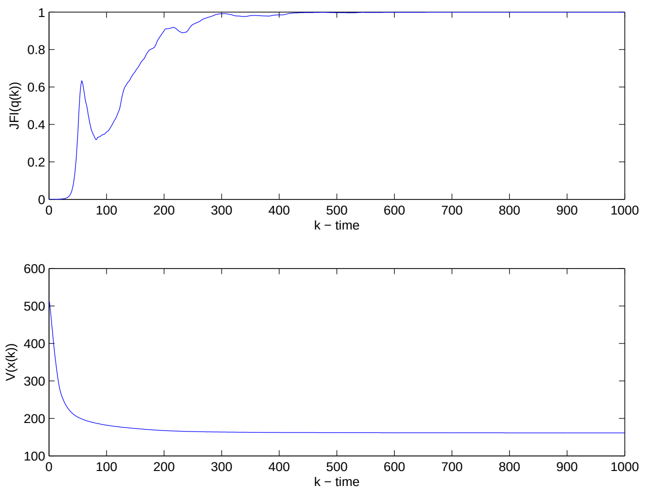

Fig. 2. $D_i \equiv 0$: evolution of $JFI(\mathbf{q}(k))$ and $V(\mathbf{x}(k))$.

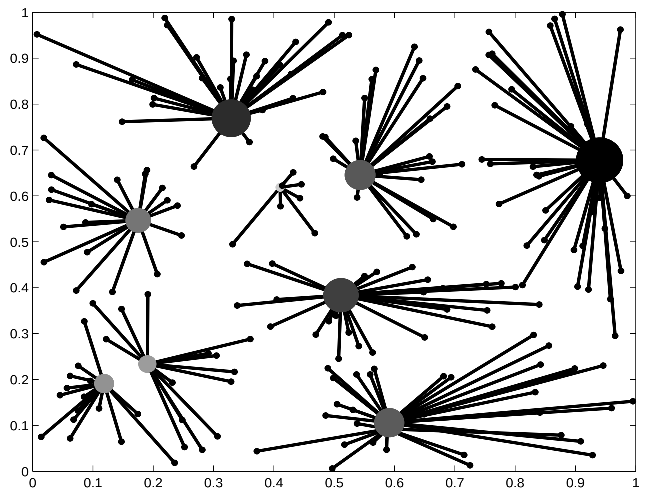

Fig. 3. $D_i \equiv 0$: steady-state. Each line represents a job-server pair with thickness proportional to $(x_j^{(i)})^*$. The diameter of each circle is μ_i^*.

N	number of servers
M	number of jobs
μ	the aggregate capacity
μ_i	capacity at server i
q_i	performance indicator at server i
d_j	demand of job j
$x_j^{(i)}$	fraction of job j served by server i
y_i	traffic intensity at server i
C_{ij}	cost of serving a unit of job j at server i
$D_i(y)$	cost of using server i with demand y
JFI	Jain's fairness index
η	gain parameter
λ	load

TABLE I
SYMBOL MAP

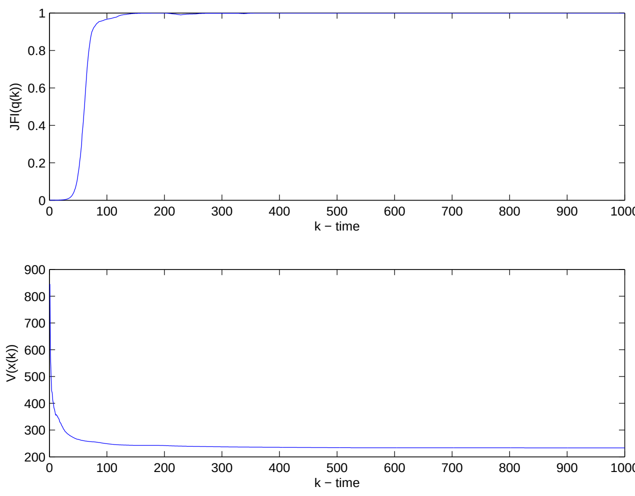

Fig. 4. $D_i(y) = y(y - \frac{\mu}{N})^+$: evolution of $JFI(\mathbf{q}(k))$ and $V(\mathbf{x}(k))$.

desired steady state values. Figure 3 depicts the steady-state results: each grey circle represents a server with a diameter proportional to its steady-state capacity. Each line represents a job-sever allocation, with the thickness proportional to the appropriate steady-state value of $x_j^{(i)}$.

Case 2. $D_i(y) = y(y - \frac{\mu}{N})^+$. This choice of cost function enforces an increasing cost when arrival demand passes[3] μ/N (so called $1/N$-rule, [19]). Then the jobs are balanced between few (close) servers to minimize the overall cost. The dynamics of the cost function $V(\mathbf{x}(k))$ as well as $JFI(\mathbf{q}(k))$ is depicted in Figure 4. Figure 5 depicts the steady-state results. As we can see the diameters of the circles representing the steady-state server-capacities are approximately equal (meaning that $D_i(y_i^*)$ is very close to zero and that overall cost V is dominated by the first part of the expression (4)). Each line corresponds to one job-server pair with its thickness being proportional to the steady state value of $x_j^{(i)}$.

B. Dynamic demands

In this simulation we use the same job and server locations with the same cost of job-server allocation: C_{ij} being the

[3]The notation means that $D_i(y) = 0$ for $y - \frac{\mu}{N} < 0$ and $D_i(y) = y(y - \frac{\mu}{N})$ otherwise.

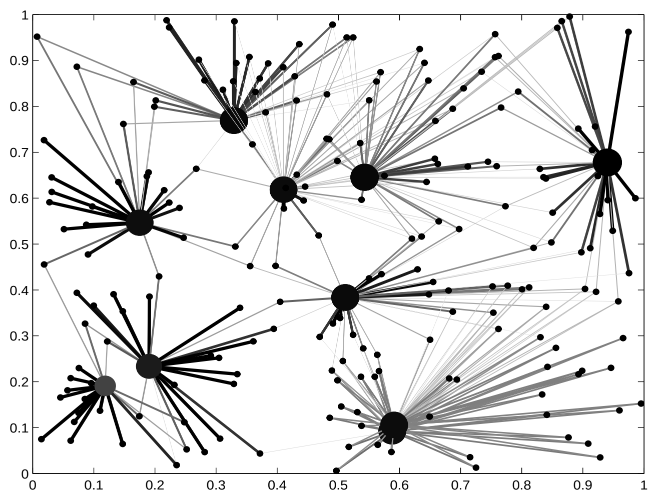

Fig. 5. $D_i(y) = y(y - \frac{\mu}{N})^+$: steady-state. Each line represents a job-server pair with thickness proportional to $(x_j^{(i)})^*$. The diameter of each circle is μ_i^*

euclidian distance between job j and server i. We vary the per job demand-intensity depending on the position of the job. Jobs that are located in the left half of the unit square (ie. with x-coordinate smaller than 0.5) start with the unit demand then slowly grow by 50% then remain constant for a while and then drop to the 50% od the initial load and remain constant until the end of simulation. The demand intensity of the jobs from the right half of the square follow the complementary trend depicted in Figure 6 (top).

We present the results for the simple demand cost functions used in previous simulations $D_i(y) = y(y - \frac{\mu}{N})^+$. The JFI evolution is depicted in Figure 6 (middle). One can notice that after initial transient phase, JFI converges to 1 and remains close to 1 throughout the course of simulation, indicating that per-server performance remains approximately uniform. The cost $V(\cdot)$ evolution, depicted in Figure 6 (bottom) shows a stable behavior with small oscillations though transient changes in the workload pattern.

Figure 7 illustrates the final configuration of the demand distribution. Since, in the last part of the simulation, the demand in the right half-square is approximately 3 times bigger than the demand in the left half-square, we can see many jobs from the right allocating the (parts of) demand to the left-half-square servers in order to balance the load and minimize the aggregate cost.

IV. Implementation issues

Comment 2: In our model, we assume that a "job" can be split in arbitrary manner. This is justified because we use term "job" for all requests coming from one domain (say, one autonomous system) having the same cost of using servers (C_{ij}). However, the extension of this fluid approximation to more realistic discrete job-partitions is an open problem.

Comment 3: In the model presented above it is assumed that subgradient algorithm is executed in a distributed manner [5] and the DNS mechanism is then used as a feedback mechanism to signal which server should be used by end-users.

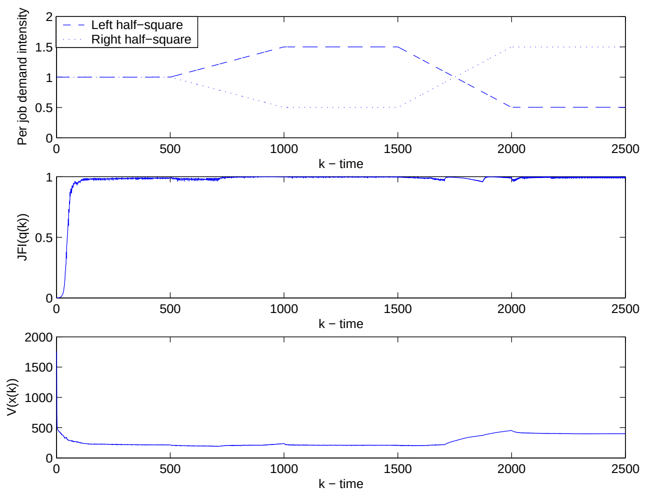

Fig. 6. Dynamic demands. Top: per job demand intensity. Middle: Evolution of $JFI(\mathbf{q}(k))$. Bottom: Evolution of $V(\mathbf{x}(k))$.

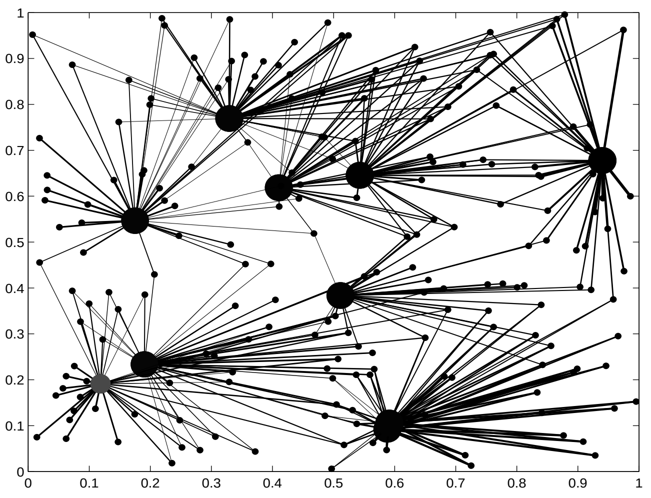

Fig. 7. Dynamic demands: final configuration. Each line represents a job-server pair with thickness proportional to $(x_j^{(i)})^*$. The diameter of each circle is μ_i^*

Comment 4: Communication between servers is performed via small UDP packets. Each of those packets should contain a field for measuring the performance at the particular server, as well as some control overhead to ensure that if a loss of a communication packet occurs no local limiter gains or loses extra capacity, and that the capacity constraint (2) is not violated.

V. Summary

Issues related to service reliability, service availability, and fault tolerance, have encouraged many service providers in the Internet to shift from traditional centric services to cloud based services. This trend appears to be a dominant mechanism for ensuring robustness of internet services with many “big players”, such as Google, Yahoo!, Akamai, Amazon, already offering a suit of cloud-based services.

Pricing, usage control, and resource allocation of cloud based services represent important technical challenges for the networking community. Load balancing, standard technique for “fair” resource control has been recently challenged by the distributed rate limiting paradigm that strives to minimize the cost in the infinite available bandwidth context. In this paper we tackle the problem of usage control without infinite bandwidth assumption. The proposed solution is simple, easy to implement and has very low computation and communication overhead.

VI. Acknowledgements

This work is supported by the Science Foundation Ireland grant 07/IN.1/I901.

References

[1] Amazon Simple Storage Service(S3):http://aws.amazon.com/s3.
[2] A. Berman, R. Plemmons. ”Nonnegative matrices in the mathematical sciences”. SIAM, 1979.
[3] D. Bertsekas, R. Gallager. “Data Networks”. 1987.
[4] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah. “Gossip algorithms: Design, analysis and applications”. In Proceedings of IEEE Infocom, 2005
[5] S. Boyd, L. Xiao, A. Mutapcic. “Subgradient methods”. Lecture notes of EE392o, Stanford University, 2003.
[6] D. F. Carr. “How Google works”. Baseline Magazine, July 2006.
[7] G. Carraro, F. Chong. “Software as a service (SaaS): An enterprise perspective”. MSDN Solution Architecture Center, Oct. 2006.
[8] J. Dilley et al., “Globally Distributed Content Delivery”. IEEE Internet Computing, vol. 6(5), 2002.
[9] D. Hinchcliffe. “2007: The year enterprises open thier SOAs to the Internet”. Enterprise Web 2.0, Jan. 2007.
[10] M. Huang. “Planetlab bandwidth limits”. Available online: http://www.planet-lab.org/doc/BandwidthLimits.
[11] A. Jain, J. M. Hellerstein, S. Ratnasamy, D. Wetherall. “A wakeup call for internet monitoring systems: The case for distributed triggers”. In Proceedings of HotNets-III, 2004.
[12] R. Jain. “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”. John Wiley and Sons, INC., 1991.
[13] R. Mahajan. “How Akamai works?”. Online: http://research.microsoft.com/ ratul/akamai.html
[14] S. Meyn. “Control techniques for the complex networks”. Cambridge University Press, Cambridge, 2008.
[15] M. Mitzenmacher. “The Power of Two Choices in Randomized Load Balancing”. IEEE Transactions on Parallel and Distributed Systems, vol. 12(10), 2001.
[16] Y. Nesterov, A. Nemirovsky. “Interior Point Polynomial Methods in Convex Programming”. Studies in Applied Mathematics, 1994
[17] H. X. Nguyen, D. Figueiredo, M. Grossglauser, P. Thiran. “Balanced Relay Allocation on Heterogeneous Unstructured Overlays”. In Proceedings of IEEE Infocom 2008, Phoenix, AZ, USA.
[18] A. Odlyzko. “Internet pricing and the history of communications”. Computer Networks, vol. 36, 2001.
[19] B. Raghavan, K. Vishwanath, S. Rambhadran, K. Yocum, A. Snoeren. “Cloud Control with Distributed Rate Limiting”.In Proceedings of ACM SIGCOMM 2007.
[20] Z.S. Rui, N. McKeown. “Designing a Predictable Internet Backbone with Valiant Load-Balancing”. In Proceedings of IWQoS 2005.
[21] R. Srikant. ”Internet congestion control”. Control theory, 14, Birkhäuser Boston Inc., Boston, MA, 2004.
[22] R. Stanojevic, R. Shorten. “Fully decentralized emulation of best-effort and processor sharing queues”. In Proceedings of ACM Sigmetrics 2008.
[23] R. Stanojevic, R. Shorten. “Distributed rate limiting in open-loop enviorments”. Technical report. Available online: http://www.hamilton.ie/person/rade/DR$L_$open.pdf
[24] F. Wuhib, M. Dam, R. Stadler. “Decentralized Detection of Global Threshold Crossings Using Aggregation Trees”. Computer Networks, vol. 52(9), 2008.