Joint Consolidation and Service-Aware Load Balancing for Datacenters

Chuan Pham, *Student Member, IEEE*, Nguyen H. Tran, *Member, IEEE*, Cuong T. Do, Eui-Nam Huh, *Member, IEEE*, and Choong Seon Hong, *Senior Member, IEEE*

***Abstract*—Workload consolidation is an efficient approach to reduce the power consumption of datacenters, meanwhile load balancing can reduce the datacenter's user delay. Despite complexities of 1) the coupling between consolidation and load balancing methods for server allocation, and 2) the heterogeneity of server configurations, we address the joint consolidation and service-aware load balancing problem to minimize the operation cost of datacenters. We first formulate the joint optimization problem, which is NP-hard. We then solve this problem using the Gibbs sampling method. Furthermore, to improve the computation of our approach, we propose the JCL algorithm that combines Gibbs sampling and the ADMM method for parallel and distributed calculations. Simulation results also validate that our method not only reduces the power consumption and delay cost, but also balances the workload in heterogeneous servers.**

***Index Terms*—Consolidation, live migration, cloud computing.**

I. Introduction

CONSOLIDATION is one of the crucial resource management techniques that helps reduce the operational cost of datacenters. Several consolidation techniques [1], [2] have been proposed to alleviate the critical problems in datacenters and cloud providers such as high energy costs and low server utilization. Consolidation paradigm is considered as a method to dynamically control the number of active servers by liberating spare servers in order to reduce the power consumption [3]. For example, in Fig. 1a, the virtual machine (VM) workload is compacted into a small number servers, and redundant servers are released. However, existing consolidation methods suffer delay cost due to servers' high utilization [1]–[3].

On the other hand, *load balancing* is necessary to guarantee the quality of services (QoS) in cloud providers. Based on the demand from users, this policy distributes the requests evenly to servers, as depicted in the example of Fig. 1b. Even though load balancing mitigates the overloaded problem to reduce the delay cost, it has no impact on the energy efficiency of the datacenter [4].

The disadvantages of either using only a consolidation or load balancing policy can be alleviated by designing joint consolidation and load balancing features that complement each other. In the example in Table I and Fig. 1, using a joint consolidation and load balancing method not only significantly reduces the total power consumption, but also mitigates the delay cost to all servers. Especially, the joint consolidation and uniform load balancing in Fig. 1c, where the workload is distributed evenly to all heterogeneous servers with different service rates, is worse than the joint consolidation and service-aware load balancing in Fig. 1d, where the workload is distributed proportionally with respect to heterogeneous servers with different service rates. This example calculates the total operation cost based on the delay cost of an M/GI/1 queueing model [5] and the power consumption model of a datacenter [6], which are elaborated in (2) and (3), respectively.

The example illustrates a potential optimization design of the coupling between load balancing and consolidation in order to reduce the power consumption and the delay cost of datacenters. Especially, considering a datacenter with *heterogeneous* servers and various arrival rates of workload in different time periods, the design of a joint consolidation and load balancing mechanism that can properly dispatch the workload to a carefully controlled number of active heterogeneous servers is a critical challenge.

To overcome these problems, we first formulate a joint consolidation and service-aware load balancing (JCL) problem as a combinatorial optimization problem, which is NP-hard. Then, we propose the JCL algorithm based on Gibbs sampling method in order to solve the JCL optimization problem. Combining the advantages of the Gibbs sampling method and the alternating direction method of multipliers (ADMM) method, we decouple the original problem into sub-problems and propose a parallel and distributed algorithm that can eliminate the computation burden at the controller to achieve an optimal solution.

II. System Model and Problem Statement

A. The VM Workload Model

We assume that time is discrete periods and we study the system for one specific period. We consider a cloud provider running cloud services on a set of $\mathcal{M}$ servers and a subset of active servers $\mathcal{M}_a$. The datacenter deploys a front-end server to receive user requests, called a dispatcher, as shown in Fig. 2. In a specific scheduling period, the total amount of incoming workload at the dispatcher is λ_s. The amount of workload distributed from the dispatcher to server i is λ_i, satisfying the following load balance constraint:

$$\sum_{i \in \mathcal{M}_a} \lambda_i = \lambda_s, \forall \lambda_i \geq 0. \tag{1}$$

Further, physical server i, depending on the configuration (such as CPU, memory, network bandwidth, etc.), can have k_i slots that paralelly serve VMs in a round-robin manner

Manuscript received August 9, 2015; revised October 4, 2015; accepted November 2, 2015. Date of publication November 18, 2015; date of current version February 12, 2016. This work was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-(H8501-15-1015)) supervised by the IITP (Institute for Information and communications Technology Promotion). The associate editor coordinating the review of this paper and approving it for publication was M. Naeini. *(Corresponding author: Choong Seon Hong.)*

The authors are with the Department of Computer Engineering, Kyung Hee University, Yongin 446-701, Korea (e-mail: pchuan@khu.ac.kr; nguyenth@khu.ac.kr; dtcuong@khu.ac.kr; johnhuh@khu.ac.kr; cshong@khu.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2015.2501402

TABLE I
EXAMPLE OF THE OPERATION COST OF THE DATACENTER UNDER DIFFERENT MANAGEMENT POLICIES

$E^I = 200$		$E^P = 400$		$\lambda_s = 40$
Good consolidation, bad load balancing				
	Server 1 $\mu_1 = 6$	Server 2 $\mu_2 = 5$	Server 3 $\mu_3 = 4$	Total operation cost (power and delay cost)
Utilization	98%	4.2%	Off	
Power cost	397	284	0	980
Delay cost	295	4	0	
Good load balancing, bad consolidation				
Utilization	44%	53%	67%	
Power cost	289	307	333	949
Delay cost	4	6	10	
Joint consolidation and uniform load balancing				
Utilization	67%	80%	Off	
Power cost	333	360	0	723
Delay cost	10	20	0	
Joint consolidation and service-aware load balancing				
Utilization	83%	60%	Off	
Power cost	367	320	0	720
Delay cost	25	8	0	

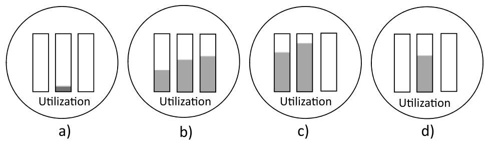


Fig. 1. Illustrating the need for joint *consolidation* and *load balancing*. a) Good consolidation, bad load balancing; b) Good load balancing, bad consolidation; c) Joint consolidation and uniform load balancing; d) Joint consolidation and service-aware load balancing.

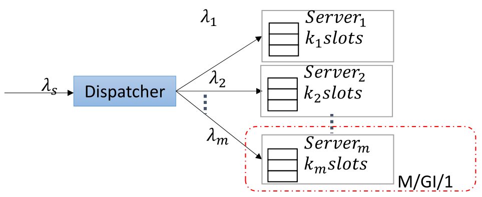


Fig. 2. The VM allocation in the datacenter.

(such as processor sharing). Therefore, assuming that the workload arrivals are Poisson, each server is modelled as M/GI/1 Processsor Sharing queue [5] with a service rate μ_i to serve VMs in one slot. In different time periods, k_i and μ_i can be varying, but $k_i \mu_i$, the capacity of server i, is fixed. We can consider k_i as the number of virtual servers of queueing system inside the physical server i. The utility formula is as follows $U_i = \frac{\lambda_i}{k_i \mu_i}$ [5].

B. The Datacenter Cost Model

As mentioned above, we focus on reducing the total operation cost and balancing the workload among servers. Therefore, in this paper, we tackle two important decisions for the datacenter: (i) determining a subset $\mathcal{M}_a$ that satisfies the amount of workload λ_s, and (ii) assigning the amount of workload λ_i to server $i, \forall i \in \mathcal{M}_a$. These issues can be modeled as the operating costs incurred by a) power consumption cost of the active set $\mathcal{M}_a$ and b) the delay costs in the datacenter.

Power consumption cost. The power consumption of the processing servers depends on the number of active servers and the utility of each server [6] as follows:

$$\alpha \sum_{i \in \mathcal{M}_a} (E_i^I + (E_i^P - E_i^I) U_i), \tag{2}$$

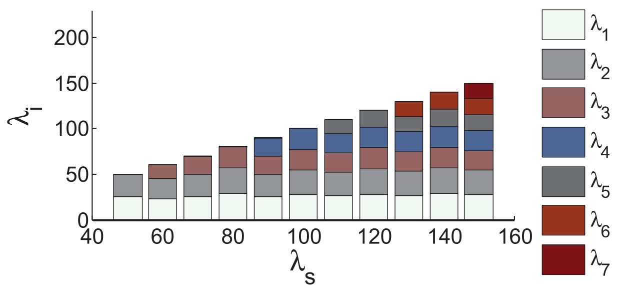


Fig. 3. Allocation scheme of the JCL algorithm.

where α is the price that converts the power to a monetary term, E_i^I is the idle server power, and E_i^P is the fully utilized server power of server i.

Delay cost. After identifying $\mathcal{M}_a$, the controller also accounts for delay cost, which was not considered in many existing works [1]–[3]. The delay cost is modeled as the M/GI/1 queueing [5]. The total delay cost of $|\mathcal{M}_a|$ active servers is as follows:

$$\theta \cdot \sum_{i \in \mathcal{M}_a} \frac{\lambda_i}{\mu_i - \frac{\lambda_i}{k_i}}, \tag{3}$$

where θ is the price that translates the delay to a monetary term.

Combining the two cost models above gives the following total operation cost $C(\lambda_i, \mathcal{M}_a)$ of the datacenter:

$$C(\lambda_i, \mathcal{M}_a) = \sum_{i \in \mathcal{M}_a} \alpha (E_i^I + (E_i^P - E_i^I) U_i) + \frac{\theta \lambda_i}{\mu_i - \frac{\lambda_i}{k_i}}. \tag{4}$$

In our work, the term "service-aware" is to emphasize that we appropriately distribute VMs to active servers based on the service rates in the heterogeneous servers.

Note that: the values of α and θ parameters are the monetary terms that are used to convert different units of power consumption cost and delay cost to calculate the total cost.

C. The Joint Consolidation and Service-Aware Load Balancing Problem (JCL)

Given the cost models above, we consider the joint problem to choose the routing policy λ_i and the subset active servers $\mathcal{M}_a$ in the datacenter. This is captured by the following optimization problem, named JCL problem:

$$\underset{\lambda_i, \mathcal{M}_a}{\text{minimize}} \quad C(\lambda_i, \mathcal{M}_a) \tag{5a}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{M}_a} \lambda_i = \lambda_s, \forall \lambda_i \geq 0. \tag{5b}$$

The JCL problem is a combinatorial optimization problem, which is NP-hard in general. We note that the JCL problem can be reduced to a simpler problem of VM consolidation without load balancing, which is shown to be a NP-hard bin packing problem [7].

III. JOINT CONSOLIDATION AND SERVICE-AWARE LOAD BALANCING

A. JCL: A Gibbs-Sampling-Based Algorithm

Gibbs sampling is a stochastic optimization method that can achieve the global optimal solution by probabilistically transitioning among possible states in the solution space [8]. The procedure of the JCL algorithm is as follows.

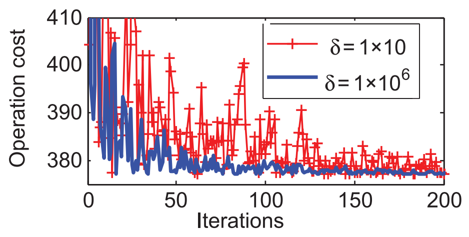


Fig. 4. Convergence of the JCL algorithm.

- **Calculate the current state**. At first, all servers are ordered based on the service rate μ_i and the capacity k_i ($\mu_1 k_1 \geq \mu_2 k_2 \geq \ldots \geq \mu_{|\mathcal{M}|} k_{|\mathcal{M}|}$). Then, the controller calculates the optimal value C^* of the JCL problem with the current subset $\mathcal{M}_a^*$ and λ_i^*. In the initialization step, the controller randomly chooses the subset of active servers $\mathcal{M}_a^* = \mathcal{M}_a'$ that satisfies the constraint:

$$\lambda_s \leq \sum_{i \in \mathcal{M}_a} \mu_i k_i, \tag{6}$$

  and a feasible λ_i^* (i.e., line 1).
- **Calculate the new state**. To move on the next state, the JCL algorithm randomly chooses the new subset $\mathcal{M}_a'$, satisfying (6). Then, it solves the JCL problem to obtain the new λ_i' and optimal value C' (i.e., line 2).
- **Move to the new state with probability** p. Based on the transition state of the Gibbs-sampling method, JCL moves to the new state with probability p (i.e., lines 3, 4). The parameter $\delta > 0$, referred to as the tunable smoothing parameter, is used to control exploration versus exploitation (i.e., the degree of randomness). As δ increases, the JCL algorithm becomes more greedy and chooses a new solution with a greater probability if it is better than the current solution (i.e., $C^* \leq C'$) [8].
- **Keep the current state with probability** $1 - p$. If the new state gives a worse optimal value than the current C^*, the controller will keep the current state (line 4).
- **Convergence**. The transition from one active server set combination to another depends only on the current state and is irrelevant to previous states. Thus, it converges to a steady state (i.e., obtaining the minimum value in JCL) after a finite number of transitions.
  As $\delta \to \infty$, the algorithm JCL converges to the globally optimal solution with a probability of 1. However, as δ becomes large, JCL is more greedy and takes more iterations to converge [8].

Based on the chosen active set $\mathcal{M}_a$, the controller turns off all servers not in this set. If one server is removed from the active set, it will not receive the request from the dispatcher (i.e., $\lambda_i = 0, \forall i \notin \mathcal{M}_a^*$) and will be switched off when all VMs release resources [1], [2] (or the controller can use the live migration technique to return all running VMs to the dispatcher).

The computational cost of JCL algorithm basically follows the complexity of the Gibbs sampling method which depends on the smooth parameter δ, since each iteration the JCL algorithm solves the subproblem (7) by a distributed method ADMM.

B. ADMM-Based for Subproblem (7)

With the heterogeneous servers in the datacenters, the computation of sub-problem (7) in Step 2 of the JCL algorithm is a burden to the controller. To tackle this problem, we parallelize and decentralize the computation using ADMM method [9]. The well-known distributed method can be applied to solve the optimization problem. It alternatively optimizes part of the objective with one block of variables in order to reach the optimum with fast convergence.

Algorithm 1. Joint Consolidation and service-aware Load balancing algorithm (JCL)

1. Initialization: Order servers, randomly choose a subset of active servers $\mathcal{M}_a'$ based on (6), and set $\mathcal{M}_a^* \leftarrow \mathcal{M}_a'$. Choose a feasible λ_i^* to compute the optimal value C^* of the JCL problem.
2. Obtain λ_i' and the corresponding optimal value $C' = C(\lambda_i', \mathcal{M}_a')$ by solving the following sub-problem:

$$\begin{aligned} \text{minimize} \quad & C(\lambda_i, \mathcal{M}_a') \\ \text{subject to} \quad & \sum_{i \in \mathcal{M}_a'} \lambda_i = \lambda_s, \\ & \lambda_i \geq 0, \forall i \in \mathcal{M}_a'. \end{aligned} \tag{7}$$

3. Compute the transition probability [8]

$$p = \frac{\exp(\delta C')}{\exp(\delta C') + \exp(\delta C^*)}. \tag{8}$$

4. With probability p, the controller sets $\lambda_i^* \leftarrow \lambda_i'$, $\mathcal{M}_a^* \leftarrow \mathcal{M}_a'$, and $C^* \leftarrow C'$. With probability $1 - p$, the controller keeps the current state.
5. Choose a new active set $\mathcal{M}_a'$ that satisfies (6).
6. Return to Step 2 until the stopping criteria is met.

In the JCL algorithm, when an active set $\mathcal{M}_a'$ is determined in each iteration, the sub-problem (7) in Step 2 can be reformulated as follows:

$$\begin{aligned} \underset{\lambda_i}{\text{minimize}} \quad & \sum_{i \in \mathcal{M}_a'} \left(\alpha \frac{\lambda_i}{\mu_i - \frac{\lambda_i}{k_i}} + \theta a_i \lambda_i\right) \\ \text{subject to} \quad & \sum_{i \in \mathcal{M}_a'} \lambda_i = \lambda_s, \\ & \lambda_i \geq 0, \forall i \in \mathcal{M}_a', \end{aligned} \tag{9}$$

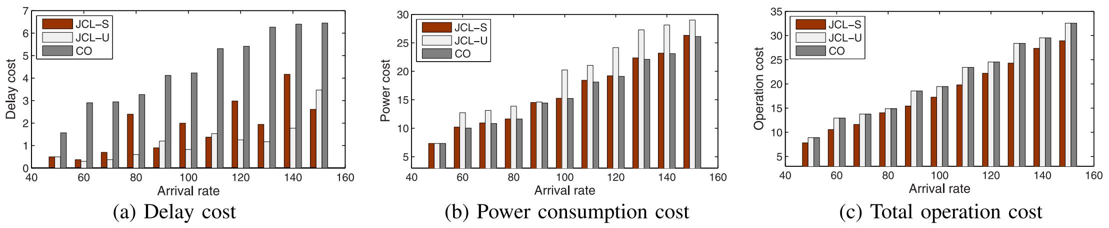
where $a_i = \frac{E_i^P - E_i^I}{k_i \mu_i}$.

However, we cannot apply ADMM directly to problem (9). Therefore, we introduce a set of auxiliary variables $y_i = \mu_i - \frac{\lambda_i}{k_i}$; replace $f_i(y_i) = \frac{\alpha(\mu_i k_i - y_i k_i)}{y_i}$, $g_i(\lambda_i) = \theta a_i \lambda_i, \forall i \in \mathcal{M}_a$; and reformulate problem (9) as follows:

$$\begin{aligned} \underset{y_i, \lambda_i}{\text{min.}} \quad & \sum_{i \in \mathcal{M}_a'} f_i(y_i) + g_i(\lambda_i) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{M}_a'} \lambda_i = \lambda_s, \\ & y_i = \mu_i - \lambda_i / k_i \geq 0, \forall i, \\ & \lambda_i \geq 0, \forall i \in \mathcal{M}_a'. \end{aligned} \tag{10}$$

The augmented Lagrangian of the above problem (10) can be formed by introducing an extra L_2 norm term $\|y_i - u_i + \lambda_i/k_i\|_2^2$ to the objective:

$$\begin{aligned} L_p = \sum_{i \in \mathcal{M}_a'} & f_i(y_i) + g_i(\lambda_i) + \gamma_i(y_i - u_i + \lambda_i/k_i) \\ & + \rho/2(y_i - u_i + \lambda_i/k_i)^2. \end{aligned} \tag{11}$$



(a) Delay cost (b) Power consumption cost (c) Total operation cost

Fig. 5. Comparison between only consolidation (CO), joint consolidation with uniform load balancing (JCL-U), and joint consolidation with service-aware load balancing (JCL-S).

The main operations of ADMM for problem (10) are as follows:

y-minimization: Each iteration $t+1$, the y-minimization step involves solving the following problem:

$$\begin{aligned} \min_{y_i}.\quad & \sum_{i\in\mathcal{M}'_a} f_i(y_i) + \gamma_i^t y_i + \rho/2(y_i^2 - 2y_i\mu_i + 2y_i\lambda_i^t/k_i) \\ \text{s.t.}\quad & y_i \geq 0, \forall i. \end{aligned} \tag{12}$$

λ-minimization: Simultaneously, the datacenter can solve the sub-problem to obtain λ_i based on (11):

$$\begin{aligned} \min_{\lambda_i}.\quad & \sum_{i\in\mathcal{M}'_a} g_i(\lambda_i) + \gamma_i^t y_i^{t+1} + 2y_i^{t+1}\lambda_i/k_i \\ & + \rho/2(\lambda_i^2/k_i^2 + 2y_i^{t+1}\lambda_i/k_i - 2\mu_i\lambda_i/k_i) \\ \text{s.t.}\quad & \sum_{i\in\mathcal{M}'_a} \lambda_i = \lambda_s, \\ & \lambda_i \geq 0, \forall i \in \mathcal{M}'_a. \end{aligned} \tag{13}$$

In a practical environment, the controller can implement JCL algorithm using $|\mathcal{M}'_a|$ threads to solve simultaneously the problem (12). Each thread obtains y_i^{t+1} and broadcasts it to the controllers. Similarly, we use the parallel computation in the λ-minimization problem (13) to obtain λ_i^{t+1}.

Dual update: During each iteration, the controller updates the dual variable γ as follows:

$$\gamma_i^{t+1} = \gamma_i^t + \rho(y_i^{t+1} - \mu_i + \lambda_i^{t+1}/k_i). \tag{14}$$

IV. Numerical Results

Settings: We use log files from web servers in the private cloud system, traced from 10 servers and 50 VMs, to evaluate the efficiency of our model.Based on the log files, we set and order the capacities ($k_i\mu_i$) of servers in range from 15 to 30. In terms of the monetary term weights, both α and θ are set to 0.01. We also set the number of slots of each server $k_i = 5$. To measure the power consumption, we assume $E_i^I = 200$W, $E_i^P = 400W$. In this work, we consider the interval duration that can adapt to a practical settings of typical datacenter: e.g., 30 minutes to hours [2].

Results: We evaluate the performance of the proposed JCL algorithm with arrival rates from 50 to 150. Fig. 3 presents the distribution of VM requests and the consolidation ability of our method (when the arrival rate increases, the number of active servers also increases and vice versa). Since the JCL algorithm chooses the number of active servers to serve the current workload, all servers not in the active server set can be switched off to reduce power consumption.

In addition, we evaluate the convergence of the JCL algorithm using 50 servers with uniform service rates μ_i in the range [3, 6]. Fig. 4 plots the convergence of the objective values with different δ values, which shows to match the property of the Gibbs-sampling method (with a larger δ, the convergence of JCL algorithm is faster).

We further demonstrate the efficiency of the proposed JCL algorithm in optimizing the power consumption and the delay cost in the datacenter. Using only the consolidation approach, Fig. 5a shows the highest delay cost. The joint consolidation and uniform load balancing can obtain the minimum delay cost, but it suffers the highest power consumption as shown in Fig. 5b. Balancing between the delay cost and the power consumption, our method results in the minimum total operation cost, as depicted in Fig. 5c.

V. Conclusion

In this paper, we propose the joint consolidation and service-aware load balancing mechanism for datacenters. We formulate the coupling consolidation and load balancing problem in a datacenter and apply the Gibbs sampling and the ADMM method to decompose the prior problem. We then present the JCL algorithm to adjust the active server set for reducing power consumption and to obtain the optimal distribution of VMs to active servers. Further, our algorithm can obtain the fast convergence using the parallel computation method. The analysis and simulation results show that our proposed system can be applied to the scheduling function of datacenters.

References

[1] A. Beloglazov and R. Buyya, "Openstack neat: A framework for dynamic consolidation of virtual machines in openstack clouds—A blueprint," Cloud Comput. Distrib. Syst. (CLOUDS) Lab., Dept. Comput. Inf. Syst., The University of Melbourne, Melbourne, Australia, 2012, http://www.cloudbus.org/reports/OpenStack-neat-Blueprint-Aug2012.pdf.

[2] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: A consolidation manager for clusters," in *Proc. ACM SIGPLAN/SIGOPS*, 2009, pp. 41–50.

[3] W. Vogels, "Beyond server consolidation," *Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[4] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Proc. Compilers Oper. Syst. Low Power (COLP)*, Barcelona, Spain, 2001, vol. 180, pp. 182–195.

[5] L. Kleinrock, "Queueing Systems, Volume II: Computer Applications," Apr. 1976, 576 pp., ISBN: 978-0-471-49111-8.

[6] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH*, vol. 35, no. 2, 2007, pp. 13–23.

[7] S. Martello and P. Toth, *Knapsack Problems*. Hoboken, NJ, USA: Wiley, 1990.

[8] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York, NY, USA: Springer, 2013.

[9] H. Xu and B. Li, "Joint request mapping and response routing for GEO-distributed cloud services," in *Proc. IEEE INFOCOM*, 2013, pp. 854–862.