Analyzing Security and Energy Tradeoffs in Autonomic Capacity Management

Ítalo Cunha, Itamar Viana, João Palotti, Jussara Almeida, and Virgílio Almeida
Department of Computer Science, Federal University of Minas Gerais
Belo Horizonte, Brazil, 31270-000
{cunha, itamar, palotti, jussara, virgilio}@dcc.ufmg.br

Abstract—Capacity management of a hosting infrastructure has traditionally focused only on performance goals. However, the quality of service provided to the hosted applications, and ultimately the revenues achieved by the provider, depend also on other aspects, such as security and energy constraints. This paper extends our self-adaptive SLA-driven capacity management solution to capture, in an unified framework, key performance and cost tradeoffs that arise when operating under security attacks and energy constraints. A number of scenarios and strategies based on dynamic SLA contracts are designed to help uncover, via simulation experiments, the main tradeoffs, considering both the provider’s interests (i.e., revenues) and the customer’s interests (i.e., legitimate throughput, response time distribution and costs). Finally, we also assess the cost-effectiveness of our framework under highly variable application service times.

I. INTRODUCTION

Capacity management of a hosting infrastructure (i.e., data center) has been traditionally developed as a set of techniques to achieve performance goals. However, the quality of service provided to customers (i.e., the owners of hosted applications) and ultimately the revenues achieved by the infrastructure provider depend on other aspects as well.

One such aspect relates to the impact of security attacks on the cost-effectiveness of alternative capacity management decisions. Despite the plethora of defense/recovery techniques available [1]–[3], security attacks, especially those aiming specific applications [3]–[5], can still cause great financial losses [1], [6]. During such an attack, illegitimate requests admitted into the system consume available resources, thus impacting the performance of concurrent legitimate requests to the same application. The provider, on its side, is also penalized, as it does not capitalize over the resources allocated to accomodate the illegitimate traffic.

Energy costs and constraints can also add extra challenges to the capacity management task, especially for heterogeneous applications sharing a complex multi-tier platform. Up to 10% of the budget of current enterprises is spent in energy, and this fraction is expected to increase even further [7]. In such scenario, new tradeoffs arise to couple with the costs and benefits of resource allocation, in particular when energy restrictions are planned and thus must be enforced.

Overall, many issues challenge the management of large and complex data centers. Cost-effective solutions can not afford to address only one issue at a time. The need to handle these issues compounds the management task, demanding tools and models capable of handling them all jointly.

Nevertheless, most previous capacity management strategies [8]–[12] focus on performance issues only. In particular, we have proposed a self-adaptive capacity management framework that dynamically allocates capacity among applications so as to maximize the provider’s revenues [11], [12]. Our framework combines a pricing model, based on Service Level Agreement (SLA) contracts, a queuing-based performance model, and an optimization model. It provides guarantees on throughput and response time tail distribution, and captures the tradeoffs of current multi-tier virtualized platforms [12].

Energy-related costs have been considered by some previous capacity management schemes [10], [13]. However, they are combined with simplified assumptions of single-tier platforms or guarantees only on average performance targets, which may not be accurate for heterogeneous and variable workloads. Moreover, most prior efforts towards addressing security issues aim at improving robustness ([3] and references within), and are disconnected from the capacity management goal.

Therefore, our main goal is to build on our previous efforts [11], [12] towards providing a simple and yet cost-effective framework to address jointly three issues that are key to capacity management, namely, service level (i.e., performance), security and energy constraints, under the same pricing model.

To the best of our knowledge, this is the first study that jointly considers these three aspects in capacity management.

More specifically, we aim at capturing the most primary issues related to performance, security and energy constraints that are key to capacity management in a unified framework, aiming at shedding some light into the following questions: (1) what are the main performance and cost tradeoffs for managing capacity under security attacks and energy constraints?, and (2) given a hosted application is under a security attack (or the infrastructure is under energy constraints), what are the cost-benefit tradeoffs of using dynamic SLA contracts, both from the provider’s and customer’s perspectives? Towards this goal, we extend our framework to capture primary aspects related to security attacks and energy consumption, as well as to implement alternative strategies based on adaptive SLA contracts. We run simulations, with synthetic and realistic workloads, to understand the main tradeoffs considering the provider’s revenues, the customer’s legitimate throughput (i.e., goodput) and response time distribution, as well as the amount...
charged to the customer for each legitimate request served. Complementary, we also assess the sensitivity of our framework to one of its key assumptions, namely, that application service times are exponentially distributed.

Our main findings are: (1) when an application is under attack, our new security-aware framework significantly increases the provider’s revenue by shifting capacity between applications accordingly, at the cost of great penalties to the victim’s goodput; (2) these penalties can be reduced (and even eliminated) if the victim agrees to pay extra for each legitimate request served; (3) alternatively, the penalties may also be reduced if the victim agrees to relax the response time SLA, although the effectiveness of this strategy depends on both the original SLA and the relaxation factor; (4) beyond maximizing resource utilization to save energy, our energy-aware framework captures key tradeoffs during energy restrictions, turning off more resources at the costlier tiers and favoring applications with lighter demands; (5) response time SLA relaxation may also reduce the goodput degradation of applications under energy restrictions; (6) our framework is cost-effective for non-exponential service times with moderate coefficients of variation (under 3).

This paper is organized as follows. Section II briefly reviews our capacity management framework and other previous work. The extensions introduced to the framework to capture security and energy aspects are presented in Section III. Simulation results are discussed in Section IV. Section V evaluates the sensitivity of our framework to highly variable service times. Conclusions and future work are offered in Section VI.

II. BACKGROUND

A. Autonomic Capacity Management for Multi-Tier Services

We consider the case of a provider hosting third-party applications on a shared infrastructure. Application owners, i.e., the customers, sign Service Level Agreement (SLA) contracts with the provider. Each hosted application may be composed of multiple request types, characterized by different workloads and resource demands. We refer to each such request type as an application class, and assume the provider hosts $N$ classes. The shared platform is composed of $K$ tiers. Each tier runs a virtualization mechanism [14], [15] that allows the creation of independent VMs. Thus, each class runs on $K$ dedicated VMs, one at each tier. After being served at tier $j$, a class $i$ request leaves the system with probability $p_{i,j}$ ($i = 1..N, j = 1..K, p_{i,K} = 1$) or enters tier $j + 1$.

In this target environment, the capacity allocation problem is defined as the determination of the fraction of the physical capacity of each tier $j$ to each class $i$ that maximizes the provider’s business goal. Such decision has to be made in light of estimates of each class expected workload, performance targets and pricing contracts (i.e., SLAs) as well as system configuration, and should be revisited in case of changes.

In [11], [12], we propose a self-adaptive capacity management framework which works as follows. Periodically, the capacity manager takes predictions of the future workloads for hosted classes, their SLA requirements, per-tier average service times and routing probabilities $p_{i,j}$, as well as system parameters to compute the capacity allocation for the next interval. It also computes the fractions of the expected request rate from each class that can be accepted into the system. To that end, it relies on a workload forecasting module [16] to monitor incoming workloads and predict each class expected workload for the next interval, as well as on an admission control mechanism [17] to enforce per-interval accepted request rates. Fig. 1 shows an iteration from one tier’s perspective.

The capacity manager combines a pricing model and a system performance model into an optimization model to solve the capacity allocation problem. The main characteristics of each model component are summarized next.

1) Pricing Model: The pricing model specifies QoS requirements and hosting costs and, thus, captures the SLA contracts. Our pricing model is based on two operation modes, normal and surge. For the former, the model defines $X_i^N$, the valid throughput (i.e., service level) that is expected to satisfy class $i$ requirements most of the time. In case of SLA violations, the provider must refund the customer $c_i$ for each unit of throughput below $X_i^N$, provided class $i$ arrival rate is high enough. For the surge mode, the model defines $X_i^S \geq X_i^N$, the maximum valid throughput up to which the customer agrees to pay extra ($r_i$ per unit of valid throughput above $X_i^N$) to accommodate occasional load peaks.

The valid throughput includes all accepted requests served with a response time that satisfies the SLA. We consider an SLA that specifies guarantees on the probability that the system response time experienced by each class $i$ request, $R_i$, exceeds threshold $R_i^{SLA}$, that is, $P(R_i > R_i^{SLA}) \leq \alpha_i$. Given $\lambda_i$, class $i$ request arrival rate, and $\lambda_i^{acc}$, the rate of requests accepted into the system, computed by the capacity manager, the provider’s revenues from class $i$, $g_i$, is given by:

$$g_i = \begin{cases} -c_i \left(\min(\lambda_i, X_i^N) - \lambda_i^{acc}\right) & \lambda_i^{acc} \leq X_i^N \\ r_i \left(\lambda_i^{acc} - X_i^N\right) & X_i^N < \lambda_i^{acc} \leq X_i^S \end{cases}$$

2) Performance Model: The analytical queuing-based performance model estimates, for each hosted class, per-tier resource utilization, system throughput and the probability of system response time SLA violations. We define $d_{i,j}$ as the average service time of a class $i$ request running on tier $j$’s full capacity, which can be estimated in a pre-production environment and inflated to capture fixed virtualization overhead [10].

The average service time of a class $i$ request at its assigned
VM on tier \( j \), \( d_{i,j} \), is then computed as \( d_{i,j} = \frac{d^*_{i,j}}{f_{i,j}} \), where \( f_{i,j} \) is the fraction of tier \( j \)'s capacity assigned to class \( i \). The effective arrival rate from class \( i \) at tier \( j \), given by \( \lambda^*_i,j \), is computed from \( \lambda^*_i \) and probability vector \( p_{i,j}, j = 1..K \). Finally, the maximum utilization planned for the VM assigned to class \( i \) at tier \( j \) is set to \( v_{i,j} \) to avoid saturation.

For each hosted class \( i \), the model assumes Poisson request arrivals with rate \( \lambda_i \), as observed in real systems \([9]\), and exponentially distributed service times at each tier \( j \), with average \( d_{i,j} \). Thus, each VM is modeled as an M/M/1 queue with FCFS scheduling discipline \([18]\), as in other previous work \([8],[9]\). Each class \( i \), in turn, is modeled as a sequence of M/M/1 queues with independent residence times.

Under these assumptions, system throughput is given by \( \lambda_{i,acc} \), and the utilization of tier \( j \) by class \( i \) is given by \( \rho_{i,j} = \lambda^*_i,j \). The probability that a class \( i \) request violates its system response time SLA \( R^*_i \), i.e., \( P(R_i > R^*_i) \), can be derived from the distribution of system response time \( R_i \). Under the assumption of M/M/1 queues, \( R_i \) follows a hypo-exponential distribution \([18]\), with parameters computed from \( d_{i,j} \) and \( \lambda^*_i \), (see Equation 1 in \([12]\)). Alternatively, a simpler approximation for \( P(R_i > R^*_i) \) based on Chebyshev’s Inequality \([18]\) can be used \([12]\). However, since the exact solution based on the hypo-exponential distribution yielded somewhat more cost-effective solutions with reasonable solution times \([12]\), we consider it as baseline in this paper.

3) Optimization Model: The pricing and performance models are combined into an optimization model with an objective function that expresses the provider’s goal of maximizing total revenues (summing Equation 1 for all hosted classes). Moreover, constraints are added to specify limits on the accepted request rates, per-tier allocated capacity, VM utilizations and effective request rates as well as to express the SLA tail distribution response time requirement for each class. A detailed description of the optimization model, as well as a discussion on optimality issues, solution times and results from an extensive evaluation are presented in \([12]\).

B. Related Work

Capacity management has been addressed by many previous studies. Some of them focus on admission control strategies \([19]\), others on shared capacity allocation among hosted applications \([8],[15]\). Several previous works \([9]–[12]\) combines both schemes into autonomic capacity management frameworks. However, most of these efforts lack one or more of the following desirable characteristics: business-oriented pricing models \([14],[15]\), probabilistic guarantees on performance targets \([8],[15]\), and modeling of multi-tier platforms. In particular, \([9]–[11]\) combines SLA-based pricing models with queuing-based performance models to derive guarantees on response time tail distribution, aiming at maximizing the provider’s revenues. However, these studies target single-tier platforms. Multi-tier platforms are addressed in \([8]\), but the proposed solution is not coupled with a pricing model and focuses on average performance targets. Overall, previous capacity management schemes, including ours (presented in Section II-A), focus mainly on performance, leaving out security and energy issues that may also impact their cost-effectiveness.

The Green Grid \([20]\) is one initiative towards energy-efficient infrastructures endorsed by major vendors. Moreover, hardware-based methods towards energy savings are studied in \([21]\). In software, a few capacity management frameworks have addressed energy costs. In \([10]\), energy costs are included in an optimization model that maximizes the provider’s revenues. In \([13]\), fine-grained energy costs are considered, allowing dynamic voltage scaling in the processors.

A list of references on DDoS attacks and defense mechanisms, including methods based on IP backscatter, wavelet-based anomaly detection and signature-based intrusion detection, is given in \([3]\). A framework to classify DoS attacks using spectral analysis is given in \([22]\). Other defense mechanisms deployed on the network \([1]\) and at the victim \([2]\) to filter malicious traffic are also available. However, none of these efforts are embedded into the capacity management context.

This work extends our capacity management framework for multi-tier platforms \([12]\) to include energy costs and capture key tradeoffs arisen by security attacks and energy constraints.

III. CAPACITY MANAGEMENT UNDER SECURITY ATTACKS AND ENERGY CONSTRAINTS

Our goal is to design simple scenarios that allow us to (1) uncover primary tradeoffs in managing capacity of a shared infrastructure under security attacks and energy constraints, and (2) evaluate the cost-effectiveness of adaptive SLA contracts. These scenarios are presented below as alternative capacity management solutions, built from our original framework.

A. Security Attacks

We focus on security attacks that target a specific application, by means, for instance, of a flooding of illegitimate requests (e.g., flooding of HTTP requests \([5]\) and spams), which, despite consuming resources, do not generate revenue. The impact of these attacks on the infrastructure and applications is of particular interest because, even though they might be detected\(^1\), illegitimate requests can not be individually blocked before being processed, as they look like legitimate ones. In fact, such attacks have been recently reported as causes of great financial losses \([1],[5],[6]\). Moreover, such attacks make for an interesting case for our capacity manager to take alternative measures to minimize overall degradation\(^2\).

In order to capture the primary impact of an attack to application class \( i \), our framework is modified as follows. The total request rate from class \( i \), \( \lambda_i \), is broken into \( \lambda_i^+ \), the rate of legitimate requests, and \( \lambda_i^- \), the rate of illegitimate requests, i.e., \( \lambda_i = \lambda_i^+ + \lambda_i^- \). Moreover, since illegitimate requests can not be identified and, thus, may be admitted into the system, class \( i \) throughput is broken into legitimate (i.e., goodput), given by \( \lambda_{i,acc} \times (\lambda_i^+/\lambda_i) \), and illegitimate,
given by $\lambda_{i,acc}^s \times (\lambda_i^- / \lambda_i)$. Thus, the service of a fraction of class $i$ legitimate requests implies extra costs to the provider, as some of the illegitimate requests will also be admitted into the system, consuming resources. This creates interesting scenarios to explore the tradeoffs between provider’s interests and customer’s interests. Moreover, we also assume legitimate and illegitimate requests have the same average per-tier service times ($d_{i,j}$), as would be expected from attacks that try to mimic user behavior [5]. The impact of attacks with heterogeneous demands, such as semantic attacks [4], are not captured in this model, but we expect the general tradeoffs discussed in Section IV to hold under those types of attacks, thus providing insights for cost-effective management decisions. Extending the model and the solutions presented below to consider more sophisticated attacks is left for future work.

Under these assumptions, the provider’s revenues from class $i$, computed only over its goodput $\lambda_i^{good} = \lambda_{i,acc}^s \times \lambda_i^- / \lambda_i$, are:

$$g_i^s = \begin{cases} 
-c_i \left[ \min(\lambda_i^s, X_i^N) - \lambda_i^{good} \right] \\
\left( \lambda_i^{good} - X_i^N \right) X_i^N - \lambda_i^{good} \leq X_i^N \\
\left( \lambda_i^{good} - X_i^N \right) X_i^N < \lambda_i^{good} \leq X_i^N
\end{cases}$$

(2)

We consider four alternative capacity management solutions, built from variants of our baseline framework:

- **Attack-Oblivious (AO):** the provider is unaware that application (class) $i$ is under attack, i.e., $\lambda_i^s$ is unknown. Capacity management is done using our original framework, which relies on estimates of revenues given by Equation 1. However, actual revenues are computed using Equation 2.

- **Attack-Aware (AA):** capacity management is performed using the extended framework described in this section.

We also consider two other strategies based on adaptive SLA contracts. In face of an attack, the victim customer may agree to either pay extra for each legitimate request served within the pre-defined (fixed) SLA, or, for fixed costs, relax the response time target. Adaptive SLAs can be beneficial to both customers, who will have more legitimate requests served, and to providers, who will receive higher revenues or have additional flexibility, and thus may help achieve a better compromise in case of conflicting interests. The next two solutions apply these strategies to the extreme, allowing us to assess their main tradeoffs and potential benefits:

- **Adaptive Cost (AC):** the victim customer agrees to pay an amount for each legitimate request served that is inflated by the attack weight (i.e., $\lambda_i^- / \lambda_i$). In other words, it pays the original fixed cost for each request served, legitimate or not.

- **Adaptive $R_{i,SLA}$ (S-AR):** the victim agrees to relax $R_{i,SLA}$ by a factor proportional to the attack weight, i.e., $R_{i,SLA,\ast} = R_i^{SLA}(1 + w_i^s (\lambda_i^- / \lambda_i))$, where $w_i^s$ is an inflation factor.

**B. Energy Constraints**

In face of energy constraints, the goal of the capacity manager is to reduce the capacity allocation, thus saving energy by increasing VM utilizations, while still meeting SLA targets. Moreover, even under normal energy conditions, it may be more beneficial to the provider to turn some of the resources off, if costs associated with the energy consumed by them is not payed off by revenues from customers.

We extend our framework to define $e_j$, the cost per time unit of operation of tier $j$ at its full capacity. The total energy cost (per time unit) associated with tier $j$ is thus $e_j \sum_{i=1}^{N} f_{i,j}$. This model makes two assumptions: (1) capacity allocation is continuous, (2) energy cost increases linearly with allocated capacity. Although simplified, this model allows us to capture primary tradeoffs introduced by energy costs, especially for systems with homogeneous components. Previously used in [10], this is the first effort to apply this model for multi-tier platforms with possibly heterogeneous costs.

The original expression for revenues is then extended as:

$$g_i^s = \sum_{j=1}^{K} (e_j f_{i,j}) + g_i$$

(3)

In order to explicitly capture energy restrictions, we add a constraint on the total capacity allocated across all tiers to the optimization model. That is, we set $\sum_{j=1}^{K} s_j \sum_{i=1}^{N} f_{i,j} \leq C$, where $C$ is the total available capacity and constants $s_j$ are used to normalize tier capacities.

We consider the following solutions for evaluation:

- **Energy-Oblivious (EO):** capacity is managed using the original framework [12], which does not capture energy costs.

- **Energy-Aware (EA):** capacity management uses the extended framework with the aforementioned changes.

- **Adaptive $R_{i,SLA}$ (E-AR):** suppose the energy consumed by the infrastructure must be reduced by a fraction $S$, reducing the total available capacity to $C_r = C(1 - S)$. In this case, the owner of application class $i$ may agree to relax its response time SLA by a factor proportional to $S$ to achieve higher throughput under the reduced allocation. That is, $R_{i,SLA,\ast} = R_i^{SLA}(1 + w_i^s S)$, where $w_i^s$ is an inflation factor.

We do not consider an adaptive cost scheme to avoid unfairness. If a customer pays extra during a period of energy restriction, he will certainly receive a higher fraction of resources, whereas others will experience service degradation. Although beneficial to the provider, this would not be fair to the other customers. In the case of security attacks, on the other hand, the extra cost charged to the victim is to its own benefit, and does not incur penalties on the other customers.

**IV. EVALUATION**

This section shows simulation results to illustrate the main tradeoffs that arise when managing capacity under security attacks (Section IV-B) and energy constraints (Section IV-C), taking into account the provider’s revenues as well as the customers’ goodput, response time distribution and cost per legitimate request served. The workload and system configurations used in our evaluation are described in Section IV-A.

---

3In a real deployment, the amount of resources (e.g., servers) to be turned off could be set to the nearest discrete value. Moreover, in case of tiers with heterogeneous components and non-linear energy costs, the framework could operate by taking the resource with the worst capacity per energy ratio first. Finally, fixed energy costs, although not explicitly modeled, should not impact the main tradeoffs, as they represent constants in the objective function.
The simulator periodically calls the capacity manager to compute, analytically, the optimal capacity allocation. The queues at each tier are then simulated, with service demands inflated according to the allocation, in order to capture workload fluctuations. Simulation results presented are averages of 5 runs with standard deviations under 2% of the means.

### A. Workload and System Parameters

We consider two scenarios describing the workload and system configurations. Both scenarios consist of infrastructures with two tiers ($K = 2$), which requests always visit (i.e., $p_{i,1} = 0, \forall i$). We assume application classes have homogeneous SLA parameters, and set $\alpha_i = 0.1, c_i = 1, r_i = 0.5, \nu_{i,j} = 0.95$, and, unless otherwise noted, $R_i^{SLA} = 20 \sum_{j=1}^{K} d_{i,j}^*$. The workload is assumed to be known a priori, and the capacity manager is executed whenever (legitimate and illegitimate) request rates change. The impact of this simplification on revenue is shown to be under 11% in [12]. Revenues are expressed considering SLA payments and thus are non-negative.

**Scenario 1** consists of two classes with requests arriving according to the step-like non-homogeneous Poisson processes shown in Fig. 2, which cover different patterns of competing synthetic workloads. Each step lasts 1000 seconds. Average service times, $d_{i,j}^*$, shown in Table I, are chosen so that the infrastructure is under-provisioned for the total load most of the time (except near step 19). Table I also shows the throughput and response time SLA parameters for each class.

**Scenario 2** is built from logs to a real Web portal, containing per-hour arrival counts for six months (01/01-30/06/06). We build realistic workloads by breaking the log into six subtraces, one per month, and taking each one as pertaining to a class. Per class arrivals are assumed to follow a non-homogeneous Poisson process with rates given by each subtrace. The infrastructure is under-provisioned with classes’ parameters given in Table I. The workloads have typical daily variations with request rates $\lambda_i^+$ shown in Table II.

### B. Security Attacks

We first discuss the most relevant results for scenario 1, augmented with an attack to class 1 at rate $\lambda_1^+$=5000 requests per second throughout simulation, i.e., an attack weight $\lambda_i^+/\lambda_1$ varying from 0.81 to 0.96. Class 2 is not under attack, and energy costs are not modeled. Fig. 3-a shows the provider’s revenues for the four capacity management solutions analyzed. Figs. 3-b and 3-c show the goodput of classes 1 and 2, respectively, with different ranges in the y axis.

The main differences between the Attack-Oblivious (AO) and Attack-Aware (AA) solutions are primarily due to AA shifting capacity from class 1 (under attack) to capitalize upon a higher goodput of class 2. AA avoids wasting capacity with illegitimate requests, thus increasing overall goodput and the provider’s revenues. The fraction of resources shifted to class 2, and thus the revenue gains of AA over AO, increase with the load on class 2. For instance, during steps 5 and 30, when class 2 workload is heavy (see Fig. 2), revenues under AA are significantly higher than under AO (Fig. 3-a), at the expense of a reduction in class 1 goodput (Fig. 3-b) and an increase in class 2 goodput (Fig. 3-c). Moreover, if the attack is too heavy (e.g., up to step 7), the victim class may be completely turned off by AA so as to maximize profits from class 2. However, if class 2 load is light, there is no benefit in shifting capacity, and both solutions yield similar results (between steps 12 and 18). Overall, AA provides significant revenue gains over AO (16% on average). From the customers’ perspective, response times and costs are unaffected, but the victim class goodput is greatly penalized (i.e., 41%) whereas class 2 benefits from extra capacity, reaching a total goodput that is 34% higher.

We now turn our attention to the adaptive SLA solutions (AC and S-AR) as strategies to reduce the penalties on the victim’s goodput. Unlike AA, the Adaptive Cost (AC) solution does not shift capacity away from the victim class, as the extra costs paid completely mitigate the impact of the attack on the provider’s revenues. In fact, Fig. 3-a shows that revenues under AC are the greatest of the four solutions, and on average 59% higher than AA. Moreover, Figs. 3-b,c show that the goodput of both classes (in particular of the victim) is as high as under AO, and 70% higher than under AA, on average. However, this comes at the expense of a cost per legitimate request served that is almost 7 times higher, on average, which may be interesting only to the most critical applications.

The S-AR solution takes advantage of the response time relaxation ($w_i=5$) to admit a larger number of class 1 requests, increasing VM utilizations. Compared to AA, it increases the victim’s average goodput in 18% with minor impact on class 2 goodput (unlike AC). In return, response time 90th percentile grows from 0.03s to 0.13s. If the increased delay is acceptable, S-AR may be a cost-effective option to applications under attacks. On the provider’s side, S-AR results in average revenues that are only 1.2% higher (Fig. 3-a).

These results are summarized in Table III, which show averages of the provider’s revenues and the customers’ cost per request and goodput. Compared to AO, AA yields greater revenues by increasing class 2 resource allocation and, thus, goodput, but the victim goodput drops significantly. AC yields the highest revenues, and the victim class has the same goodput as in AO, but at a significantly higher cost per request.

### Table I

**Application Classes’ Parameters**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>$i$</th>
<th>$d_{i,1}$</th>
<th>$d_{i,2}$</th>
<th>$R_i^{SLA}$</th>
<th>$\lambda_i^+$</th>
<th>$\lambda_i^0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Synthetic)</td>
<td>1</td>
<td>0.9 ms</td>
<td>0.6 ms</td>
<td>0.03 s</td>
<td>500</td>
<td>1200</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>12 ms</td>
<td>8 ms</td>
<td>0.4 s</td>
<td>18.4</td>
<td>36.8</td>
</tr>
<tr>
<td>(Realistic)</td>
<td>2</td>
<td>4, 6</td>
<td>8 ms</td>
<td>12 ms</td>
<td>0.4 s</td>
<td>18.4</td>
</tr>
</tbody>
</table>

### Table II

**Scenario 2 (Realistic) Workload Characteristics**

<table>
<thead>
<tr>
<th>Class</th>
<th>$\lambda_i^+$ (req/s)</th>
<th>Start Time</th>
<th>Duration (min)</th>
<th>$\lambda_i^0$ (req/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1, 24, 25</td>
<td>08h10m44s</td>
<td>10</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>0.6, 13, 27</td>
<td>17h00m00s</td>
<td>30</td>
<td>261</td>
</tr>
<tr>
<td>3</td>
<td>0.5, 18, 41</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.3, 20, 39</td>
<td>36h26m32s</td>
<td>10</td>
<td>518</td>
</tr>
<tr>
<td>5</td>
<td>0.6, 13, 25</td>
<td>08h00m00s</td>
<td>500</td>
<td>62</td>
</tr>
<tr>
<td>6</td>
<td>0.7, 18, 34</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
</tbody>
</table>
Finally, S-AR is able to increase revenues and goodput, at the cost of longer response times for the victim class (not shown).

We now analyze the impact of varying attack rates. Fig. 4 shows that the difference between AA and AO increases with \( \lambda_1^* > 0 \). This is due to idle capacity being used to serve, and capitalize upon, illegitimate requests (around step 19). The gains increase with the fraction of idle capacity available.

Next, we discuss the difference between AA and S-AR in detail. Three key points are worth mentioning. First, revenue and goodput differences depend on the attack weight. The heavier the attack, the smaller the fraction of legitimate requests served, and smaller is the benefit from relaxing the victim’s \( R_{t}^{S,SLA} \). Fig. 4 shows S-AR’s revenues converging to AA’s as \( \lambda_1^* \) increases. Goodput curves for both classes (not shown) have similar behavior. Second, the tighter the original \( R_{t}^{S,SLA} \), the greater the impact of relaxing it. Since VM utilizations and response time are related by a non-linear function, the gains from relaxing the original \( R_{t}^{S,SLA} \) are greater when it is before the knee of the curve. This is illustrated in Fig. 5, which shows the response time distributions for both AA and S-AR with original \( R_{t}^{S,SLA} \) equal to 10d (tight) and 50d (loose), where \( d = \sum_{j=1}^{K} d_{i,j} \). The distance between AA and S-AR curves for each case (tight and loose) gives an idea of the increase in utilization, and thus, goodput. The relaxation of the tight \( R_{t}^{S,SLA} \) gives more room for improvement. In fact, the average gains in the victim goodput (revenues) provided by S-AR goes from 4% (0.5%) for the loose SLA to 77% (3.1%) for the tight SLA, with no impact on class 2 goodput. Finally, all results shown are for a relaxation factor \( \nu = 5 \). Greater values of \( \nu \) lead to similar results as VM utilizations are already close to the maximum allowed (\( \nu_{i,j} \)). Lower values lead to smaller differences between S-AR and AA.

We now consider scenario 2, built from realistic workloads, augmented with 4 attacks with durations and rates \( \lambda_1^* \) derived from [22]. Attacks target classes 1, 2, 4 and 5, each with start time (from simulation startup), duration and rate shown in Table II. In order to highlight the tradeoffs due to the attacks, we report, in Table IV, average results for the total interval during which at least one class is under attack, and for the interval during the attack to class 5. For the latter, we show metrics only for the victim class. Both aggregated and per-class results illustrate the same tradeoffs identified in scenario 1. AA greatly improves revenues over AO at the cost of penalties to the victim goodput. AC eliminates the penalties at a significant increase in costs, whereas S-AR provides some gain for the victim at a compromise in response time. The gains of R-SLA are modest due to the loose original SLA.

C. Energy Constraints

Our main goal is to show key tradeoffs that arise when energy costs are included into capacity management, comparing the EA and E-AR solutions when energy savings are desirable (\( S=0 \)) or must be enforced (\( S>0 \)). The benefits of considering such costs, i.e., EA over EO, are discussed later. We show results for scenario 1, with no attacks, keeping the term goodput to refer to an application class throughput. Tiers have equal capacity (\( s_1=s_2=1 \), \( C=2 \)), and per-tier energy costs are set to a percentage of the maximum possible revenue, \( K/(d_{i,j}^1+d_{i,j}^2) \). Given a fixed total energy cost, two configurations are considered, namely, homogeneous costs fixed at 18%
(i.e., $e_1 = e_2 = 240$), and heterogeneous costs of 30% ($e_1 = 400$) and 6% ($e_2 = 80$). The tradeoffs shown hold for other costs.

We start by evaluating the EA solution for $S = 0$. Revenues, shown in Fig. 6-a, are the same for both homogeneous and heterogeneous configurations, as total energy cost is fixed. Fig. 6-b shows per-class goodput for each configuration. Class 2 goodput for the homogeneous case (omitted) is similar to class 1, shifted according to the workload. For the heterogeneous case, however, it is more cost-effective to turn off more resources from the costlier tier (tier 1), removing them from the class with heavier (local) demand (class 1). Class 1 is thus penalized, as shown in Fig. 6-b, with an 8% decrease in its goodput. Occasionally, the reduction in class 1 goodput may lead to some resources of tier 2 being shifted to class 2, favoring it. These results are summarized in Table V. Note the tier allocations (two rightmost columns), with more resources being turned off from tier 1 in the heterogeneous case.

When there is a 10% energy constraint ($S = 0.1$), revenues and goodput of both classes decrease due to the reduced capacity, as shown in Figs. 6-a and 6-c. The same tradeoffs observed for $S = 0$ hold in this case. However, in the heterogeneous configuration, EA is even more aggressive in removing tier 1 capacity from class 1. Figs. 6-b and 6-c show that class 2 goodput is less penalized by the reduction in capacity (6% versus 10% for class 1, on average). Moreover, Table V shows that, comparing the two configurations for $S = 0.1$, class 1 average goodput decreases in 5%, whereas class 2, with the lighter demand on the costlier tier, is favored with an average goodput 3% higher in the heterogeneous case.

We now consider the use of the E-AR solution, based on adaptive response time SLAs, when goodput $3\%$ higher in the heterogeneous case. Fig. 6-a, 6-b, and 6-c show that class 2 goodput is less penalized by the reduction in capacity (6% versus 10% for class 1, on average). Moreover, Table V shows that, comparing the two configurations for $S = 0.1$, class 1 average goodput decreases in 5%, whereas class 2, with the lighter demand on the costlier tier, is favored with an average goodput 3% higher in the heterogeneous case.

![Graph](image)

**Fig. 6.** Results for Energy Constraints (Scenario 1, $s_1 = s_2 = 1$, $C = 2$, $w_i^c = 25$)

**TABLE V**

<table>
<thead>
<tr>
<th>Sol.</th>
<th>Config.</th>
<th>Rev./s</th>
<th>$\lambda_{i}^{acc}$</th>
<th>$\lambda_{i}^{acc} \times f_{i}$</th>
<th>$\sum_{j} f_{j}$</th>
<th>$\sum_{j} f_{j} \times C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA</td>
<td>$S = 0$, Hom.</td>
<td>473</td>
<td>491</td>
<td>491</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>EA</td>
<td>$S = 0$, Het.</td>
<td>476</td>
<td>475</td>
<td>490</td>
<td>0.89</td>
<td>0.97</td>
</tr>
<tr>
<td>EA</td>
<td>$S = 0.1$, Hom.</td>
<td>546</td>
<td>526</td>
<td>537</td>
<td>0.87</td>
<td>0.87</td>
</tr>
<tr>
<td>EA</td>
<td>$S = 0.1$, Het.</td>
<td>541</td>
<td>491</td>
<td>461</td>
<td>0.83</td>
<td>0.91</td>
</tr>
<tr>
<td>E-AR</td>
<td>$S = 0.1$, Hom.</td>
<td>546</td>
<td>518</td>
<td>519</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td>E-AR</td>
<td>$S = 0.1$, Het.</td>
<td>549</td>
<td>486</td>
<td>548</td>
<td>0.82</td>
<td>0.86</td>
</tr>
</tbody>
</table>

**SUMMARY OF IMPACT OF ENERGY CONSTRAINTS: AVERAGE RESULTS**

We now turn our focus to performance, leaving out security issues, to assess the sensitivity of our original framework to the assumption of exponential service times. We run experiments with scenario 1, assuming per tier service times follow lognormal distributions with various coefficients of variation (CV). Exponential service times are taken as baseline. We then consider three estimates of $P(R_i > R_i^{SLA})$.
Hypoexp.:

- **Hypoexponential**: uses the hypoexponential distribution which is exact for exponential service times (M/M/1 queues).

- **Chebyshev (M/M/1)**: assumes M/M/1 queues and uses the upper-bound $P(R_i > R_i^{SLA}) \leq \frac{Var[R_i]}{E[R_i] - E[R_i]^{SLA}}$, given by Chebyshev’s Inequality [18], where $E[R_i]$ and $Var[R_i]$ are the mean and variance of system response time, which are easily computed for M/M/1 queues [12].

- **Chebyshev (M/G/1)**: Chebyshev’s Inequality is used but each tier is modeled as an M/G/1 queue. Mean and variance of response time are computed from corresponding metrics of per-tier residence times, which, in turn, are estimated from the service time distributions (lognormal) and waiting times. Waiting time metrics are estimated using the first three moments of the service time distributions, as detailed in [18]. Models based on G/G/1 queues (derived from [23]) yielded similar results with higher complexity, and are thus omitted.

Note that all three models introduce extra errors by assuming Poisson arrivals at the back-end tier. Table VI shows average revenues and probabilities of SLA violations, for various distributions and models. The very aggressive allocation decisions made by the hypoexponential model [12], yielding a probability of violations close to target $\alpha_i=0.1$ in the baseline, leaves little room to accommodate extra variance. As CV increases, the number of violations escalates and revenues drops. The more conservative Chebyshev (M/M/1) model [12] still meets the SLA for CVs under 3, with small impact on revenues. By capturing service time variability, the more complex Chebyshev (M/G/1) estimate always satisfies the SLA, at the cost of very conservative admission control decisions, and ultimately very low revenues. Similar results are obtained for other distributions of service times. Thus, in case of non-exponential distributions, our framework, using the simple Chebyshev’s bound and assuming M/M/1 queues [12], yields satisfactory results for moderate CVs (under 3). Note that the conclusions drawn in Section IV, using the hypoexponential model and assuming exponential service times, should hold also for the Chebyshev (M/M/1) model, as both yield the same tradeoffs for exponential service times [12].

### VI. CONCLUSIONS

This paper extends our capacity management framework to capture key cost and performance tradeoffs introduced by security attacks and energy constraints. Based on experiments with synthetic and realistic workloads, we reached the following conclusions. First, introducing energy and security awareness into our framework provides great revenue gains at the cost of goodput degradation, particularly for applications under attacks with little room to accommodate extra variance. As CV increases, the number of violations escalates and revenues drops. The more conservative Chebyshev (M/M/1) model still meets the SLA for CVs under 3, with small impact on revenues. By capturing service time variability, the more complex Chebyshev (M/G/1) estimate always satisfies the SLA, at the cost of very conservative admission control decisions, and ultimately very low revenues. Similar results are obtained for other distributions of service times. Thus, in case of non-exponential distributions, our framework, using the simple Chebyshev’s bound and assuming M/M/1 queues, yields satisfactory results for moderate CVs (under 3). Note that the conclusions drawn in Section IV, using the hypoexponential model and assuming exponential service times, should hold also for the Chebyshev (M/M/1) model, as both yield the same tradeoffs for exponential service times [12].

### ACKNOWLEDGMENT

This work was done in collaboration with HP Brazil R&D.

### REFERENCES


