Resource Protection in Enterprise Data Centers: Architectures and Protocols

Yesid Jarma

Thèse de doctorat

UNIVERSITÉ PIERRE ET MARIE CURIE
UPMC SORBONNE UNIVERSITÉS

École doctorale

EDITE DE PARIS
INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE

présentée par

Yesid Jarma

pour obtenir le grade de

Docteur de l'Université Pierre et Marie Curie

Protection de Ressources dans des Centres de Données d'Entreprise : Architectures et Protocoles

soutenue le 24 janvier 2012 devant le jury composé de :

Patrick Senac	Rapporteur	Professeur ISAE
Yacine Ghamri-Doudane	Rapporteur	Maître de Conference ENSIIE
Jérôme Brun	Examinateur	VP Cloud Services AtoS
Khaled Boussetta	Examinateur	Maître de Conference Université Paris 13
Pierre Sens	Examinateur	Professeur UPMC Sorbonne Universités
Yannis Viniotis	Co-directeur	Professeur North Carolina State University
Marcelo Dias de Amorim	Co-directeur	Directeur de Recherche CNRS

PhD Thesis

UNIVERSITY PIERRE AND MARIE CURIE
UPMC SORBONNE UNIVERSITÉS

Doctoral school

EDITE DE PARIS
COMPUTER SCIENCE, TELECOMMUNICATIONS, AND ELECTRONICS

presented by

Yesid Jarma

submitted in partial fulfillment of the requirements for the degree of

Doctor of Science of the University Pierre and Marie Curie

Resource Protection in Enterprise Data Centers: Architectures and Protocols

Commitee in charge:

Patrick Senac	Reviewer	Professor ISAE
Yacine Ghamri-Doudane	Reviewer	Associate Professor ENSIIE
Jérôme Brun	Examiner	VP Cloud Services AtoS
Khaled Boussetta	Examiner	Associate Professor Université Paris 13
Pierre Sens	Examiner	Professor UPMC Sorbonne Universités
Yannis Viniotis	Co-advisor	Professor North Carolina State University
Marcelo Dias de Amorim	Co-advisor	CNRS Research Director

Remerciements

Je tiens tout d'abord à remercier Patrick Sénac et Yacine Ghamri-Doudane d'avoir accepté de rapporter cette thèse. Malgré le court délai et leurs emplois du temps chargés, ils ont consacré le temps nécessaire pour lire ce manuscrit, et pour me faire des remarques pertinentes pour l'améliorer. Je remercie tout autant Khaled Boussetta, Jérôme Brun et Pierre Sens d'avoir bien voulu faire partie de mon jury.

Marcelo, mon co-directeur, m'a accueilli dans son équipe depuis 2007. C'est grâce à lui que j'ai appris le métier de chercheur. Grace à ses conseils, son optimisme et son sens de l'humour, j'ai pu mener à bien mon travail de thèse.

Par la suite, je voudrais remercier mon co-directeur de thèse, Yannis Viniotis. Son expérience, sa patience et son grand sens de la pédagogie m'ont aidé énormément au cours de cette thèse. Je voudrais également le remercier de m'avoir offert la possibilité de faire un stage chez IBM en Caroline du Nord.

J'adresse un remerciement particulier à Mehdi et Nadjet (ordre alphabétique et non de préférence) pour leur amitié et leur soutien au cours des ces années. Aussi je tiens à mentionner mes collègues de bureau (il faut savoir lequel car il y en a 3 différents) Alex, Diana, John, Matteo, Nicolas et Tiph, pour la bonne ambiance générale au bureau, et pour leur sympathie. Dans une ambiance pluriculturelle, comme celle au LIP6, je voudrais mentionner aussi Adisorn, Ahlem, Mihn, Mubashir, et Raul, pour m'avoir donné un petit aperçu de leurs cultures respectives.

Kato, tu has estado ahí para mi. Gran parte de los logros que he obtenido a lo largo de estos años te los debo a ti. Gracias por apoyarme y sobretodo aguantarme.

Finalmente, pero en primer lugar, quisiera agradecer a mis padres, Emigdio y Rosario. Largo ha sido el camino desde el niño que peleaba para ir temprano al colegio, hasta este hombre que todavía pelea para levantarse. Esta tesis no es mía sino de ustedes de principio a fin.

Résumé

Au cours des dernières années, l'apparition de l'Internet a changé la façon dont les affaires sont menées partout dans le monde. Pour rester compétitives, les entreprises ont déployé du support informatique pour les processus métiers au fil des années. Dans ce contexte, les architectures orientées service (SOA) ont émergé comme la solution principale pour l'intégration des systèmes patrimoniaux avec les nouvelles technologies au cœur des grandes organisations. Les centres de traitement de données d'entreprise qui implémentent les concepts et solutions des SOA sont normalement déployés en suivant une architecture à deux niveaux où, pour libérer les serveurs de services des tâches computationnelles intensives (e.g., l'analyse syntaxique de documents XML) et pour effectuer de la protection de ressources, ces fonctions sont déchargées dans un cluster d'appliances qui implémentent des fonctions des réseaux orientées service (SON). Dans les centres de traitement, l'accès aux services est gouverné par des contrats de garantie de services (SLA), dont le but est de protéger les ressources du centre de traitement. Actuellement, les appliances SON sont utilisées pour protéger les ressources du centre de traitement en limitant l'accès (e.g., en contrôlant le trafic) aux services.

Le provisionnement et l'optimisation de ressources sont des problèmes classiques de la gestion de la QoS. En outre, le contrôle de trafic est un problème très connu de l'ingénierie de trafic. Cependant, dans les centres de traitement orientés service le problème est fondamentalement diffèrent. Dans les réseaux classiques, les ressources protégée par la fonction de mise en conformité sont normalement la bande passante et la taille des mémoires tampon, dont les unités de mesure sont clairement définies et mesurées avec précision. Dans un centre de traitement, les métriques des ressources sont comprises pour la plupart dans un des types suivants : puissance de calcul et mémoire des serveurs d'application (*CPU* et *RAM*), capacité de stockage des serveurs de stockage (*espace en disque dur*), et la bande passante du réseau interne du centre de traitement. Une autre différence fondamentale est que, dans les réseaux dits « classiques », le contrôle de trafic a une étendue locale, puisque le trafic prend la conformité d'une connexion simple. Dans un centre de traitement, les

clients de service accèdent aux services à partir de *multiples* points d'entrée (p.ex., un cluster d'appliances SON). Ainsi, l'effet désiré est une mise en conformité « globale » du trafic. *Le défi est donc faire respecter les contrats de service en agissant localement dans chaque point d'entrée.*

Cette thèse apporte trois contributions. D'abord nous proposons DOWSS, un algorithme dynamique basé sur des crédits pour la mise en conformité de trafic *multipoint-à-point*. À la différence des approches existantes basées sur des crédits, notre approche utilise une stratégie doublement pondérée pour l'affectation de crédits, en utilisant des poids basés sur la taille des requêtes de service. L'évaluation de DOWSS montre que ses performances sont optimales puisqu'il limite le nombre de requêtes au maximum permis par le contrat de service.

Par la suite, nous affirmons que les appliances SON actuelles présentent des limitations architecturales qui les empêchent d'être utilisées efficacement pour la mise en conformité de trafic en présence d'hôtes de service *multiples*. Pour palier à ce problème, nous proposons MUST, une architecture interne pour les appliances SON appropriée pour la mise en conformité de trafic *multi-service*. L'évaluation des performances de notre approche montre qu'elle résout le problème de la mise en conformité de trafic *multipoint-à-multipoint* tout en poussant le système à être utilisé à sa capacité maximale.

Finalement, actuellement les applications sont souvent déployées dans des centres de données géographiquement distribués. Les approches existantes pour la mise en conformité de trafic, lesquelles ont étés conçues spécifiquement pour des centres de données aménagés sur un même site, présentent des problèmes liés aux latences réseau quand ils sont utilisés dans des environnements géographiquement distribués. Pour palier à ce problème, nous proposons GEODS, un approche pour la mise en conformité du trafic géographiquement distribué qui considère les délais de communication entre les entités qui forment le système. L'évaluation de ses performances montre qu'il est capable de résoudre efficacement le problème de la mise en conformité du trafic dans les environnements géographiquement distribués.

Mots-clefs

Protection de ressources, centres de données orientés service, réseaux orientés service, mise en conformité du trafic de service, points d'entrée multiples, distribution géographique.

Abstract

During the last few years, the rise of the Internet has changed the way business is conducted worldwide. To remain competitive, businesses have been implementing information technology support for business processes over the years. In this context, Service Oriented Architectures (SOA) have emerged as the main solution for the integration of legacy systems with new technologies within large organizations. Modern Enterprise Data Centers (EDCs) implementing SOA concepts and solutions are usually deployed as a two-tiered architecture where, in order to relieve service servers from the computational cost of CPU intensive tasks (e.g., XML parsing) and to perform resource protection, these functions are offloaded on a cluster of SON (Service-Oriented Networking) appliances. In EDC setups, access to services is governed by Service-Level Agreements (SLAs), which aim at protecting EDC resources. Currently, SON appliances are able to protect EDC resources by limiting the access (i.e., controlling the traffic) to services.

Resource provisioning and optimization is a classic QoS management problem. Moreover, traffic control is a well-known problem in network traffic engineering. However, in service-oriented EDC setups the problem is fundamentally different. In classic networks, the resource protected by the shaping function is typically link bandwidth and buffer space, the units of which are precisely defined and measurable. In an EDC environment, resource metrics mostly fall into one of the following types : CPU power and main memory from application servers (*CPU* and *memory*), disk storage from storage servers (*disk*), and link bandwidth on the internal EDC network (*bandwidth*). Another fundamental difference is that in "classic" networks traffic control has local scope, since traffic is in the form of a single connection. In an EDC environment, service clients access services from *multiple* entry points (e.g., a cluster of SON appliances). Thus, the desired effect is "global" shaping. *The challenge is then to enforce contracts by taking local actions at each entry point.*

The contributions of these thesis are threefold. We first propose and validate DoWSS, a dynamic credit-based algorithm for *multipoint-to-point* service traffic shaping. Contrary to existing credit-based approaches, DoWSS involves the use of a

doubly-weighted strategy for credit allocation. The evaluation results of DoWSS show that it performs optimally by limiting the number of requests to maximum possible number allowed by the client service contract.

Second, we argue that current off-the-shelf SON appliances present architectural limitations that prevent them from being used to efficiently perform traffic shaping in the presence of *multiple* service hosts. To tackle this issue, we introduce MuST, a SON Appliance architecture fit for *multi-service* traffic shaping. Our validation via simulation shows that our approach solves the multipoint-to-multipoint service traffic shaping problem while pushing the system to its maximum capacity.

Finally, current trends point to having applications located in geographically distributed EDCs. Existing traffic shaping approaches, which are designed for single-site EDCs, present issues related to network latencies when used in geographically distributed environments. To tackle this issue, we propose GeoDS, a geographically distributed service traffic shaping approach that considers in its design the communications delays between entities in the system. Our evaluation shows that our approach is able to efficiently solve the service traffic shaping problem in geographically distributed environments.

Keywords

Resource protection, service-oriented enterprise data centers, service-oriented networking, service traffic shaping, multiple entry points, geographical distribution.

Acronyms

EDC Enterprise Data Center.

ESB Enterprise Service Bus.

IaaS Infrastructure-as-a-Service.

MSA Manual and Static Allocation.

QoS Quality of Service.

SaaS Software-as-a-Service.

SAR Service Access Requirements.

SLA Service-Level Agreement.

SOA Service-Oriented Architectures.

SOAP Simple Object Access Protocol.

SON Service-Oriented Networking.

UDDI Universal Description, Discovery, and Integration.

VM Virtual Machine.

WS Web Services.

WSDL Web Services Description Language.

XML Extensible Markup Language.

Contents

Chapter 1

Introduction

DURING the past 20 years, the rise of Information Technology (IT) has changed the face of the world. In the late 1980's, a key finding by the Landmark MIT Study[45] indicated that IT had become a vital resource for competing in the global marketplace, which pushed many large organizations to consider IT an essential component of worldwide corporate strategy. As a result, in order to remain competitive, businesses everywhere have been implementing IT support for business processes over the years. Coupled to this, the emergence of the World Wide Web shifted the Internet, a network primarily used by academia and research, to a worldwide network connecting businesses and consumers everywhere. Fueled in part by these technological advances, most of the world economies moved from agriculture and manufacturing based economies, towards service economies, resulting in the appearance of new and innovative forms of service providing such as internet banking, highly efficient retail megastores, and e-commerce, among others.

1.1. A Service-Oriented Internet

Online services offered by companies all over the world often take the form of distributed software applications hosted in back-end servers. The current trend is to have business applications located in geographically distributed Enterprise Data Centers (EDCs)[27], either company-owned or outsourced, operating as private or public clouds, in which computing operations are able to switch over between sites in a transparent way, maintaining user sessions, application availability, and access to data resources. A recent study by Gartner Research[29] showed that France is one of the European leaders in terms of adoption of such hosting offerings, as 71% of the surveyed organizations had made use of Software-as-a-Service (SaaS) for one or

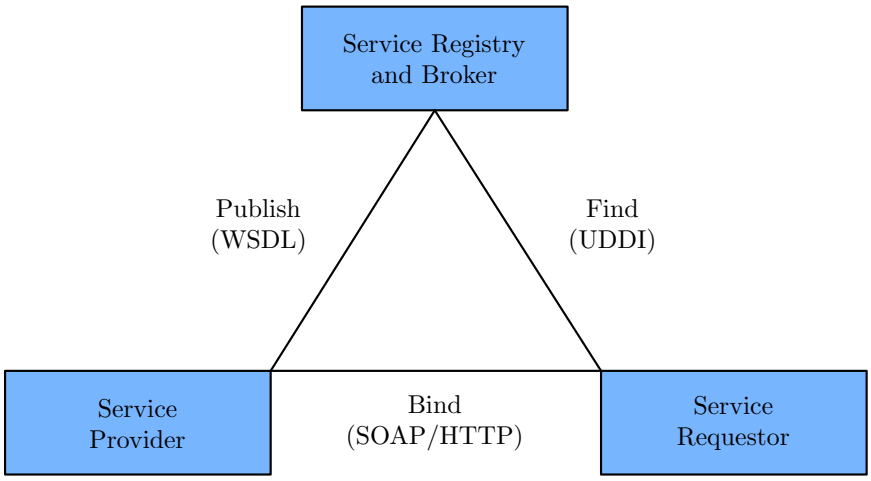


Figure 1.1: Key components of a Service-Oriented Architecture [18].

several business applications, whereas the remaining 29% were planning on doing it within the next 12 months.

Such a challenging environment raises several issues in terms of management, performance, resiliency, and security. Nevertheless, one of the biggest challenges faced by IT managers today is application integration. Because of the heterogeneity of IT systems out there, making legacy systems communicate with each other and with newer systems, across different vendors, protocols, and software, remains a difficult endeavor. Moreover, the rate of change in available hardware and software further amplifies the issue of maintaining systems that are able to adapt to business requirements. In this context, Service-Oriented Architectures (SOA)[18] have become the main solution for the integration of applications and technologies in the business domain and for the collaboration among industrial partners.

SOA is a software architecture for building applications that implement business processes or services by using a set of loosely coupled, black-box components orchestrated to deliver a well-defined level of service. It was designed to be the next generation of middleware to directly address the issues inherent in heterogeneity and change. SOA can be implemented by dint of different technologies such as Enterprise Service Bus (ESB) and Web Services (WS)[38]. The latter are software systems designed to support machine-to-machine interoperability through a set of Extensible Markup Language (XML)-based open standards, such as Web Services Description Language (WSDL)[40], Simple Object Access Protocol (SOAP)[39], and Universal Description, Discovery, and Integration (UDDI)[24] (see Fig. 1.1).

The adoption of XML-based standards, WS, and SOA has enabled the introduction of application awareness into the network fabric. For example, routing becomes

XML-oriented by being able to direct traffic based on XML content through the use of functions such as XPath routing[41]. Furthermore, offloading capabilities such as XML transformation (XSLT)[42], which changes XML content as it traverses the network, and service mediation to enable the interoperability of WS in heterogeneous environments, are now possible. Nevertheless, even though the use of XML-based standards allows easy integration with external data sources, one of the major issues preventing wider adoption of Web Services is *performance*[46]. Indeed, as the time needed to parse an XML document can take up to a few minutes[17], the response time of a Web Service is potentially large. To better satisfy business goals, service providers use middleware providing Service-Oriented Networking (SON)[7] capabilities, namely accelerated XML parsing, functional offloading, protocol integration, and content based routing.

With the integration of these new technologies, cloud-based application hosting offerings in the public domain are quickly becoming competitive. For IT managers this implies deploying and managing well provisioned EDCs, ready to handle the ever changing service loads. Indeed, cloud-based applications, needing several different CPUs to run on, potentially accessed by tens of thousands of users everyday, place enormous demands on the underlying EDC resources. In an EDC environment, these mostly fall into one of the following types: CPU power, main memory (RAM), storage capacity, request throughput, and network bandwidth, which must be shared by all the applications hosted in the EDC. Moreover, to provide reliable and scalable computing infrastructure, EDCs are specially provisioned for worst-case or busy hours. In most setups, around 45% of data center costs go to servers[15]. To make the most out of the important investments required for EDC provisioning, achieving high utilization is an important goal. Unfortunately, utilization in the EDCs can turn out to be remarkably low (e.g., 10%)[15]. Because of long provisioning time scales, the size of the investment, the uncertainty in demand, and the negative consequences of failure, conservatism leading to over-provisioning is a natural mechanism for risk management. Thus, in EDCs, resource protection targets at reducing disruptions and improving overall resource availability.

1.2. Problem Formulation: Resource Protection in Enterprise Data Centers

Resource provisioning and protection is a classic Quality of Service (QoS) problem aiming at providing service customers with service level guarantees. To better meet

the service level guarantees, and in order to protect shared EDC resources, IT managers establish Service-Level Agreements (SLAs), which define the rules for sharing EDC resources, as well as for offering service quality guarantees to customers[4]. In more detail, an SLA specifies the metrics a customer is able to use in order to monitor and verify a service contract. These metrics are laid out to match customer business objectives to quantifiable service provider performance indicators. "Classic" SLAs usually include a compromise by service providers to perform a particular activity within a clearly defined amount of time (e.g., solve high-severity issues in less than an hour) or to maintain a minimal service availability (e.g., 99.99% uptime).

There are many academic and industrial research efforts in the literature oriented specifically towards the study and implementation of resource protection mechanisms in the classic networking field. Depending on the requirements specified in a particular SLA, a number of well-known techniques aimed at sharing EDC resources, while providing service guarantees, have been developed. These include, among others, work-conserving scheduling algorithms[12;22], non-work conserving shaping algorithms[13;25;26], and load balancing techniques[10;23].

Nevertheless, in SON environments, the resource protection problem is fundamentally different. SOA are typically centralized systems in which one node executes and manages instances of one or more services. However, to address possible scalability issues, the centralized service may be replicated and the requests balanced among the replicas[21]. In general, modern EDCs are typically constructed based on a tree-like hierarchical physical topology[32] (see Fig. 1.2), and follow a two-tier logical architecture, where clients on the Internet must interact with the first tier (or preprocessing tier), in order to get access to the application instances on the second tier (or service tier).

In two-tier setups, to relieve the service tier from the computational cost of CPU intensive tasks like XML parsing and limiting the access rate to services, these functions are offloaded on a cluster of SON appliances deployed on the preprocessing tier[11]. Resource protection mechanisms available in current off-the-shelf SON appliances aim mainly at protecting CPU power on the service tier. However, the only way they able to perform resource protection, for any kind of SLA, is by limiting the number of requests per second sent to a service instance. We refer to this problem as the *service traffic shaping problem*.

A service is typically accessed from a single entry-point, and the traffic from the gateway to the service host follows a point-to-point pattern. Solutions from the "classic" packet/ATM world, as the ones mentioned above, are therefore applicable.

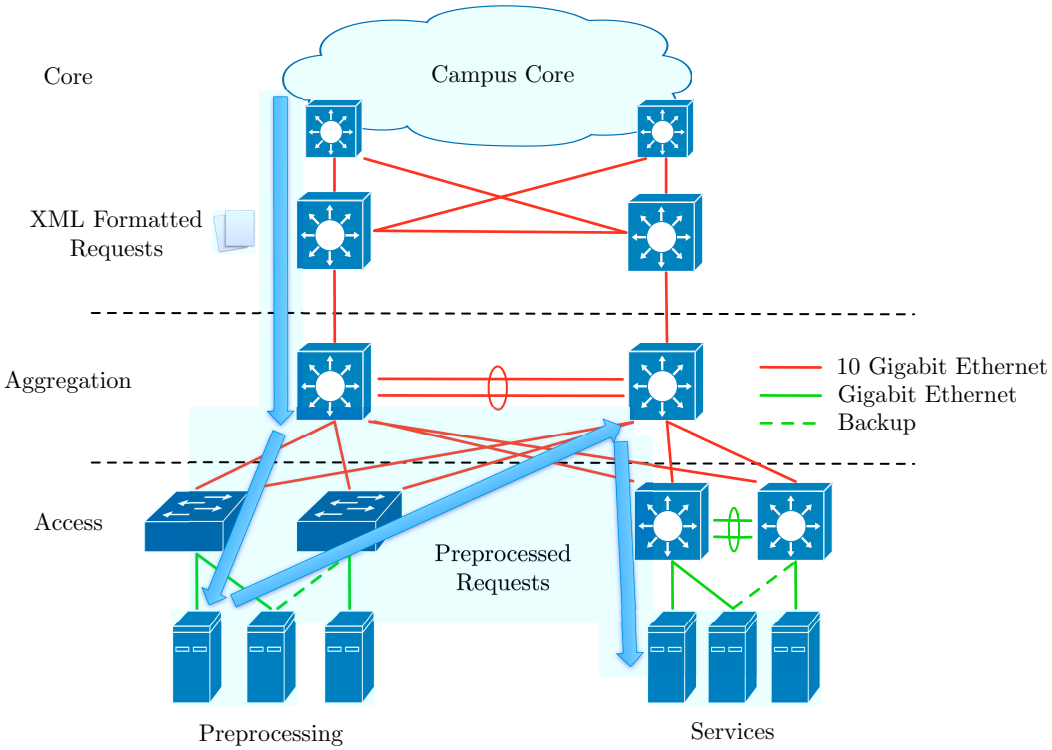


Figure 1.2: Typical Fat-Tree EDC physical topology reprinted from [32]. The flow of XML formatted service requests is represented by blue arrows.

In contrast, in SON environments clients may access *multiple* services from *multiple* entry points. In Fig. 1.2, we show three entry points at the preprocessing tier (bottom left), accessing a service deployed in 3 physical servers (bottom right). The existence of multiple entry points may be dictated by security policies (the presence of multiple security zones), robustness (fault tolerance), or performance requirements (preprocessing load is balanced on a cluster of middleware appliances); the desired effect is "global" shaping. *The challenge is therefore to enforce the SLA requirements by taking local actions at each entry point.*

In classic networks, the resource protected by the shaping function is typically link bandwidth and buffer space, the units of which are precisely defined and measurable. SLA are standardized by industrial bodies and they are very well defined. In SON, the resource protected by the shaping function is CPU processing power. Moreover, to-date *SLAs are not standardized, and therefore not precisely defined and measurable.* For example, in two-tier EDC deployments, IT managers define Service Access Requirements (SAR), aiming at protecting the service tier (located at the bottom right on Fig. 1.2 for example) from being unduly overwhelmed. These SAR definitions, in general follow the following format:

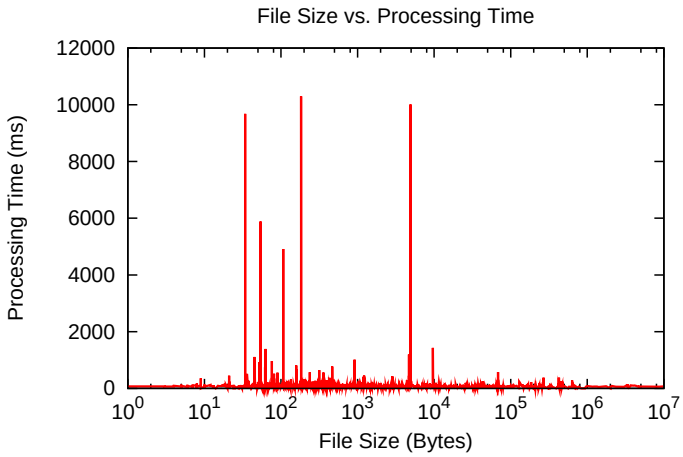


Figure 1.3: Preprocessing time of XML formatted service requests.

> SAR: "Limit the rate to a service provider to no more than X requests per second with an enforcement period of T seconds."

Where an enforcement period is a time interval during which the aggregate of requests sent to the service host by all the appliances cannot exceed $C = X \times T$. In this particular case, since "requests" are defined in units of XML requests, CPU processing time is not known exactly. In Fig. 1.3, we show the time consumption for the preprocessing of XML requests using real traffic captured in a web farm. Because of the high variability in the delays required for parsing the XML formatted requests, even though border routers at the edge of an EDC eventually perform some load balancing, this does not guarantee that the rate of requests sent to the service tier will be balanced as well.

This thesis centers specifically on efficiently solving the service traffic shaping problem between entry-point SON appliances and service instances, while considering the constraints induced by *current* SAR definitions, both of which are found in actual production systems. Nevertheless, a short discussion on the need for different SAR definitions, and the design of new system architectures more suited to solve the service traffic shaping in SON environments, is conducted in Appendix B.

1.3. Contributions of This Thesis

Given the costs and issues inherent to the implementation and provisioning of EDCs, it is imperative to find ways to optimize the overall utilization of the system. We argue that in order to efficiently perform resource protection in EDCs by shaping

service traffic, which will enable current deployments to satisfy business goals and remain cost effective, new architectures and protocols must be carefully designed. In detail, the contributions of this thesis are as follows:

1.3.1. Resource Protection in Single-Site Enterprise Data Centers

DoWSS. In most EDC deployments communication delay is not an issue as the infrastructure usually shares the same rack or is located in the same geographical site. In this kind of deployments, IT managers define service access requirements, which is the rate at which a service should be accessed, and limits it to no more than a number of service requests during an observation period. In this thesis, we first propose and validate DoWSS, a doubly-weighted algorithm for service traffic shaping from multiple access points towards a single service instance. We show via simulation that DoWSS possesses several advantages: it eliminates the approximation issues, prevents starvation and contains the rapid credit consumption issue in existing credit-based approaches.

MuST. Next, we argue that current off-the-shelf SON appliances present architectural limitations that prevent them from being used to efficiently perform traffic shaping in the presence of *multiple* service hosts. In this thesis, we introduce MuST, a SON appliance architecture fit for *multi-service* traffic shaping. We show via simulation that our approach solves the multipoint-to-multipoint service traffic shaping problem while pushing the system to its maximum capacity.

1.3.2. Resource Protection in Geographically Distributed EDC Deployments

GeoDS. Current trends point to having applications located in geographically distributed EDCs. This type of systems often induce new constraints, in addition to the ones already found in single-site systems, such as non-negligible communications delays. To tackle these new constraints, we propose and validate GeoDS, an algorithm for service traffic shaping in geographically distributed setups. We show via simulation that our approach prevents issues inherent to the presence of network latencies, and efficiently solves the service traffic shaping problem.

1.4. Outline

The remainder of this manuscript is structured as follows. In Chapter 2, we provide the necessary background before we clearly identify the problem of resource protection by service traffic shaping in Enterprise Data Centers. In Chapter 3 we propose and validate DOWSS, a doubly-weighted algorithm for service traffic shaping in single-site EDCs; we show via simulation that DOWSS possesses several advantages: it eliminates the approximation issues, prevents starvation and contains the rapid credit consumption issue in existing credit-based approaches. In Chapter 4, we identify the architectural limitations that exist in current Off-the-shelf SON which prevent them from being used to efficiently perform traffic shaping in the presence of *multiple* service hosts; then we propose and validate via simulation MUST, a SON appliance architecture fit for *multipoint-to-multipoint* service traffic shaping in two-tier EDCs, which solves the multipoint-to-multipoint service traffic shaping problem while pushing the system to its maximum capacity. After studying and proposing solutions for single-site EDCs, in Chapter 5 we identify the issues inherent to geographically distributed EDCs, namely non-neglible network latencies, before proposing and validating GEODS, a service traffic shaping algorithm which performs efficient resource protection in geographically distributed EDCs. Finally, in Chapter 6 we conclude this thesis and point to remaining open issues.

Chapter 2

Service Traffic Shaping for Resource Protection in Enterprise Data Centers

In this chapter, we introduce and motivate the problem of service traffic shaping for protecting resources in an EDC. We first provide the necessary background on Service-Oriented EDCs and service level agreements before presenting the system architecture this body of work focuses on. Next, we specify the details of the service traffic shaping problem. Finally, we present related work on this field, and discuss the shortcomings of current service traffic shaping approaches.

2.1. Service-Oriented Enterprise Data Centers

As mentioned in Chapter 1, current trends for meeting business objectives point to hosting applications in geographically distributed EDCs operating as clouds, in which computing operations are able to switch over between sites in a transparent way, maintaining user sessions, application availability, and access to data resources. Nevertheless, one of the biggest challenges is application integration. In this context, Service-Oriented Architectures (SOA) have become the main solution for the integration of applications and technologies in the business domain. SOA can be implemented by means of a variety of technologies such as Web Services, Enterprise Service Bus, and SOA middleware. The latter, frequently referred to as Service-Oriented Networking (SON) appliances, consists on using specific hardware that provides dedicated operations such as accelerated XML processing, functional offloading, service integration, and intelligent routing[7].

In this body of work, we consider this kind of deployment to be a two-tier logical system architecture as the one shown in Fig. 2.1. In this model, incoming TCP con-

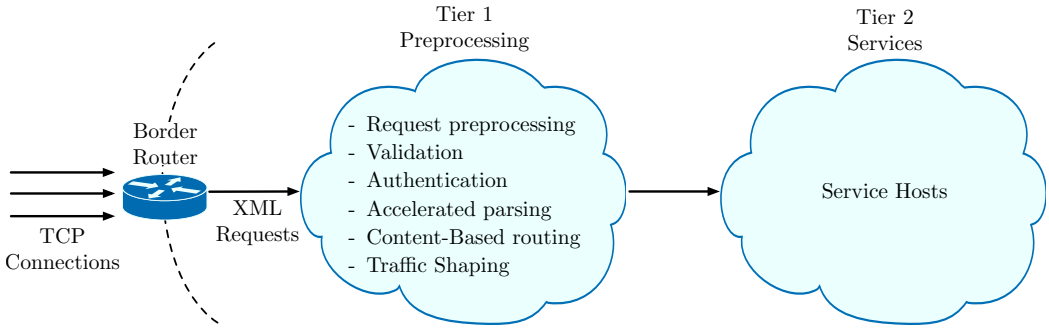


Figure 2.1: Logical architecture of a Service-Oriented Enterprise Data Center. In this model, TCP connections from outside the EDC are terminated by border routers which assemble XML-formatted requests. These requests are then forwarded to the preprocessing tier which will reroute them to the correct service in the service tier.

nections are terminated by routers at the edge of the EDC. Border routers assemble XML formatted requests to be forwarded via the EDC internal network to a cluster of SON appliances forming the *preprocessing tier*. After performing the necessary operations (parsing, authentication, validation, etc.), requests are forwarded to the correct service in the *service tier*. This kind of architecture raises interesting challenges related to resource provisioning and performance optimization, as *multiple* access points (e.g. the cluster of SON appliances) may access concurrently *multiple* service instances. Furthermore, usually in these kind of setups, besides providing the functionalities described above, SON appliances are responsible for enforcing the rules specified by a Service Level Agreement (SLA).

2.2. Service Level Agreements and Service Access Requirements

Access to services is governed by a Service-Level Agreement (SLA), which specifies the rules for sharing EDC resources, as well as for offering service quality guarantees to customers[4]. To better meet the service guarantees fixed by SLAs and in order to protect shared EDC resources, in the SON environment resource utilization metrics mostly fall into the following types:

- *CPU* is the general CPU utilization of a hosted application.
- *Memory* defines the main memory (RAM) utilization by a particular hosted service.
- *Storage* is the amount of disk storage that is allocated to a particular service.

- *Throughput* specifies the number of requests a service should receive during a given period of time.
- *Bandwidth* is the amount of bandwidth that a service uses in the EDC internal network.

Unlike their counter-partners from the "classic" networking, SLAs in SON environments are not standardized by industrial bodies. Examples taken from two major providers of Data Center colocation and hosting services further corroborate this. Even though both providers offer similar services, Verizon's SLA definitions are pointed towards specific goals in bandwidth and throughput, while Amazon's definitions dwell more on overall service availability.[31;36]

However, even in this heterogeneous environment, IT managers establish Service Access Requirements (SAR) as part of an SLA, aimed at protecting system resources. For illustration purposes, we present several examples of SAR definitions in terms of the above mentioned types. Let us first consider a government agency, say, the Internal Revenue Service (IRS), which has deployed an EDC in order to offer electronic processing of tax returns. In order to protect the IRS' EDC resources, the IT administrator defines a SAR as follows: "Limit the number of tax returns to process to no more than 1,000 over a period of 10 s". As a second example, a cloud service provider (e.g., Google) with a Software as a Service (SaaS) offering may want to protect their internal EDC network from saturation by limiting the amount of bandwidth customers use to access their applications. For this case, a SAR could be defined as: "Limit the amount of bandwidth for customer A to a maximum of 4 GB per 24-hour period". Finally, Infrastructure-as-a-Service (IaaS) providers, such as Amazon EC2, define instances for billing purposes, and for protecting their EDC infrastructure. For example, Amazon would want to limit the resources consumed by a customer's Virtual Machine (VM) by establishing the following SAR: "Client B is allowed to use a maximum of 1.7 GB of RAM, 25% of CPU power, and 160 GB of storage for its VMs".

2.2.1. SAR Enforcement in Two-tier EDCs

In two-tier setups, to relieve the service tier from the computational cost of CPU intensive tasks like XML parsing and limiting the access rate to services, these functions are offloaded on a cluster of SON appliances deployed on the preprocessing tier. Resource protection mechanisms available in current off-the-shelf SON appliances aim mainly at protecting CPU power on the service tier. However, the only

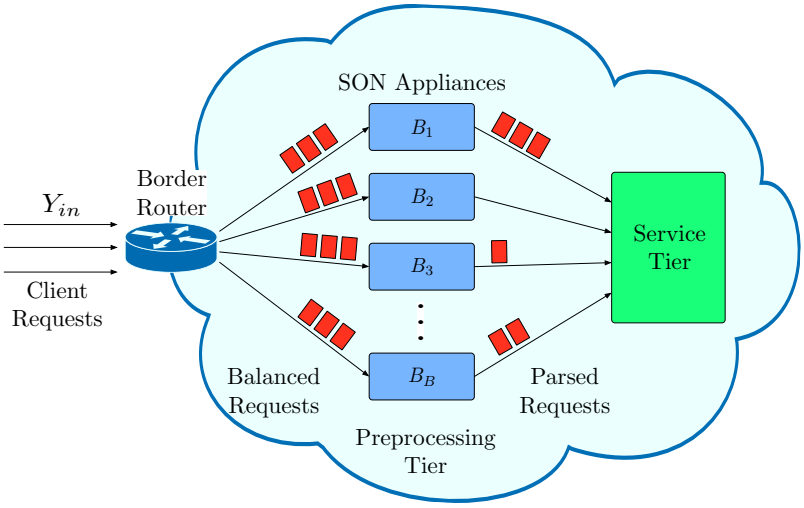


Figure 2.2: Considered system architecture illustrating the interactions between each one of the entities.

way they able to perform resource protection is by limiting the *number of requests per second* sent to a service instance. This particular metric may not be the most appropriate one for properly performing resource protection in EDCs. Indeed, since SON appliances do not perform deep content inspection of the requests to be serviced, there can be no estimates on how much CPU power or memory a particular request may require. Furthermore, even by inspecting the contents of service requests and estimating its resource needs, there is no guarantee that the estimates will be accurate enough. In the next section, we describe this problem which we refer to as the *service traffic shaping problem*, and the existing challenges when this approach is used in two-tier EDCs. Although other definitions could be provided, we still have to deal with this kind of SAR, as they are implemented in reality.

2.3. System Architecture

Service-Oriented Architectures are typically centralized systems in which one node executes and manages instances of one or more services. However, to address possible scalability issues, the centralized service may be replicated and the requests balanced among the replicas[21]. The general architecture of the considered system is depicted in Fig. 2.2. In detail, the specific functions of each architectural component are as follows:

- *Border routers*. These are the first entry point of the system. They are responsible for terminating customers' TCP connections, assembling XML-formatted

requests, and forwarding them to the preprocessing tier. Also, they are eventually in charge of distributing the service load among the appliances without any deep-content inspection.

- *Preprocessing tier*. The main building block of the preprocessing tier are SON appliances. These appliances are a specific hardware that provides simplified deployment and functions including accelerated XML processing, functional offloading, service integration, and intelligent routing. To better satisfy business goals, and to address issues such as security, fault tolerance, and performance, SON appliances are usually clustered (cf. Fig 1.2, bottom left).

- *Service Tier*. It is composed of clusters of service servers (see Fig. 1.2, bottom right) that can be application servers or storage servers. This entity processes the bulk of service requests.

A service is typically accessed from a single SON appliance; therefore, the traffic from the gateway to the service host follows a point-to-point pattern. A single entry point provides the advantage of simplified service access management. Furthermore, since point-to-point traffic shaping is a well-studied problem in the networking space, well-known solutions from packet/ATM networks can be applied.

Nevertheless, in the SON environment, clients may access *multiple* services from *multiple* entry points. The existence of multiple entry points may be dictated by security policies (the presence of multiple security zones), robustness (fault tolerance), or performance requirements (load is balanced on a cluster of SON appliances). SON appliances can implement a number of functions, which include functional offloading, service integration, and intelligent routing[11]. In addition to providing these functions, SON appliances are also responsible for controlling the rate at which client requests are sent to the service hosts. This problem is known as the *service traffic shaping* problem.

2.4. The Service Traffic Shaping Problem

Traffic shaping is a well-known classic problem in network traffic engineering[13;25]. However, in this kind of EDC setups the problem is fundamentally different. In classic packet/ATM networks, the resource protected by the shaping function is typically link bandwidth and buffer space, the units of which are precisely defined and measurable. SLAs are standardized by industrial bodies and CSC contracts are very well defined. In contrast, as described in Section 2.2, in Service-Oriented EDC

environments, the resources protected by the shaping function are CPU power, main memory, storage capacity, and link bandwidth. Moreover, since SON appliances do not perform deep content inspection of the requests to be serviced, there can be no estimates on how much CPU power or memory a particular request may require. Furthermore, SLAs contain SAR definitions which are not precisely defined and measurable.

Besides being ill-defined, current SAR included SLA agreements often overlook geographical distribution. Since most EDCs are deployed in a single site, network communications delay between entities is not a concern. However, in upcoming geographically distributed EDCs operating as geographically distributed clouds, network latency between architectural components becomes a major issue. During the remainder of this discussion we will focus on the single-service problem on single-site EDCs. A more detailed discussion on the multi-service problem and geographically distributed EDCs will be presented in Chapters 4 and 5 respectively. We are interested, in particular, in the SAR definition, which in general follows the following format (as previously shown in Chapter 1):

> SAR: "Limit the rate to a service provider to no more than X requests per second with an enforcement period of T seconds."

In this particular case, since "requests" are defined in units of XML requests, CPU processing time is not known a priori. Furthermore, this SAR does not include additional requirements such as a maximum burst size. On the other hand, in traditional networks, the parameters for implementing token buckets, for example, include, in addition to an average rate, a peak rate (which is the maximum rate at which packets can be sent in a short time interval) and a burst size (a limit for the number of packets to be transmitted in a short time interval).

In general, service instances are accessed from a single SON appliance; therefore, the traffic from the gateway to the service host follows a *point-to-point* pattern. A single entry point provides the advantage of simplified service access management. As mentioned before, since point-to-point traffic shaping is a well-studied problem in the networking space, well-known solutions from packet/ATM networks can be applied. Nevertheless, as mentioned earlier, in this kind of setups, multiple service instances may be accessed from multiple entry points. This kind of architecture raises interesting challenges related to resource provisioning and performance optimization, as *multiple* access points (e.g., the cluster of SON appliances) may access concurrently either a *single* or *multiple* service instances, as illustrated in Figs. 2.3

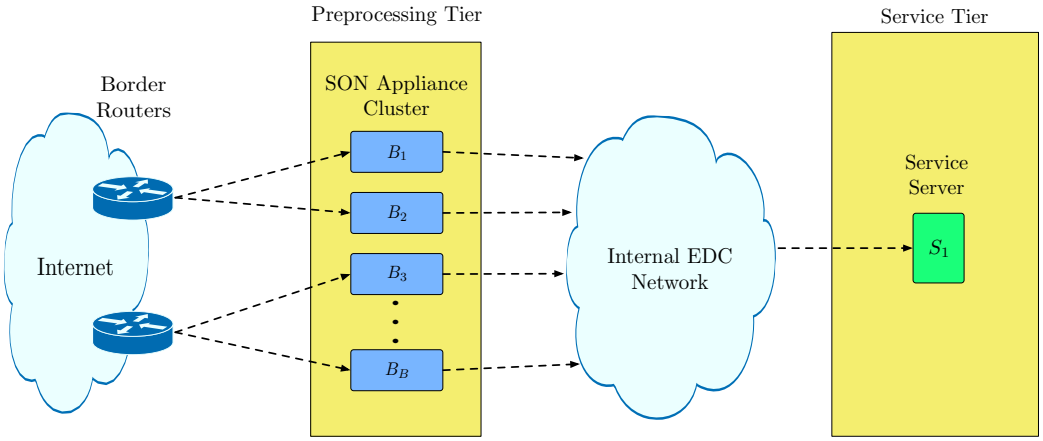


Figure 2.3: System architecture example which illustrates the case where multiple entry points, SON appliances in this case, access concurrently a single service host (*multipoint-to-point*) case.

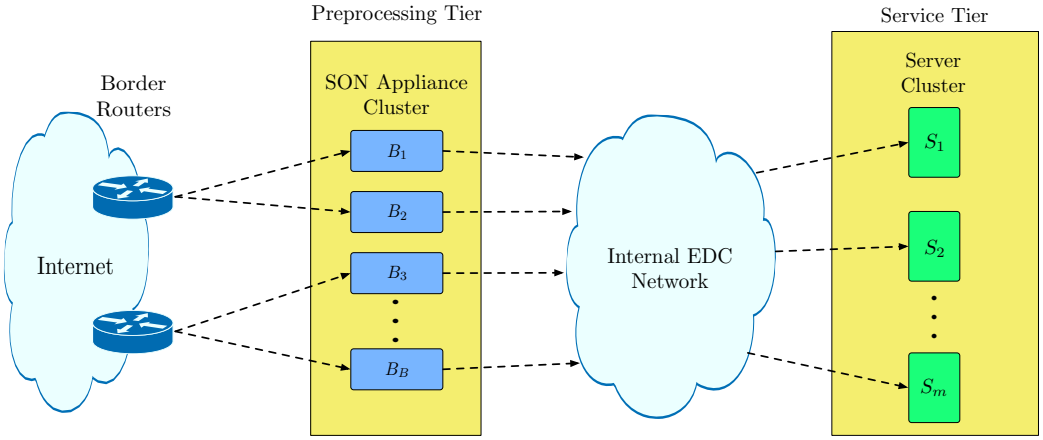


Figure 2.4: System architecture example which illustrates the case where multiple SON appliances access concurrently multiple service instances (*multipoint-to-multipoint*) case.

and 2.4 respectively. The desired effect is "global" shaping. *The challenge is therefore to enforce the traffic contract by taking local actions at each entry point.*

2.4.1. Scheduling vs. Shaping

Fig. 2.5 shows the internal architecture we consider for a SON appliance. Recall from Section 2.3 that, besides SON-related processing of XML requests, an appliance is also responsible for enforcing the SAR requirement, i.e., for limiting the rate at which processed requests are sent to the service hosts to any desired rate X, *regardless of the input rate*. In our assumed model, requests entering each appliance are placed in an input queue. A CPU performs all the SON-related tasks. A CPU scheduler determines the order in which requests from the input queue get allocated to CPU resources, by using a (work-conserving) job scheduling algorithm such as

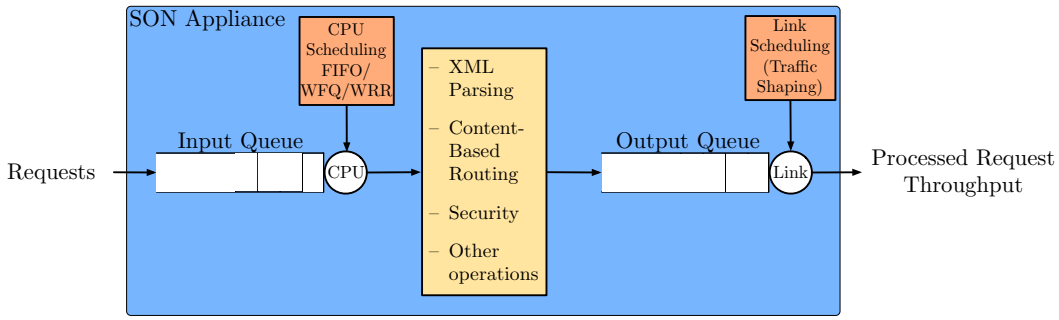


Figure 2.5: Internal Architecture of a SON Appliance showing some of its basic elements and functions.

FIFO[22] and WFQ[12]. Once the requests have been processed, they are placed in an output queue, to be transmitted to the service tier via a communication link. A link scheduler determines the order in which processed requests access the outgoing link. In our problem, we have only one class of requests, so FIFO ordering will suffice. *The SAR requirement is enforced by the link scheduler*; work-conserving algorithms are not suitable for such enforcement, since they do not limit the output rate. Non-work-conserving algorithms must be used for the control of the outgoing link. Suppose, for example, that the SAR specified $X = 2$, the input rate was 4 and the outgoing link had a capacity higher than the input rate. A work-conserving scheduling algorithm (e.g., WFQ) would not be able to enforce this SAR. For clarity and in accordance with jargon from networking environments, we label this function in Fig. 2.5 as Traffic Shaping.

2.5. Related Work

In this section we provide the general background on the service traffic shaping problem in EDCs. Further related work, specific to each of the problems we address in this thesis, will be presented in the corresponding chapters.

2.5.1. General Problem

Resource provisioning is a classic QoS management problem in EDC setups. Over the years, several works dealing with this topic have been published. In[34], the authors focus on providing an agile way for dynamically allocating enterprise data center shared resources for adapting to variations of the system's workloads by using queuing models and prediction techniques, and a system architecture based on virtual machine monitors for reducing provisioning overheads. Wang et al. propose

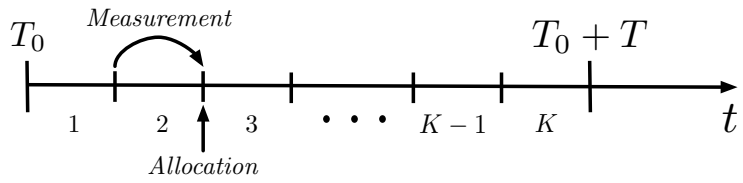


Figure 2.6: Division of the enforcement period T into K smaller subperiods.

an architecture with autonomic features along with a non-linear optimization model in order to dynamically and optimally allocate EDCs sharable resources[43]. Other bodies of work propose a different approach aiming at reducing provisioning costs while optimizing resources. Greenberg et al. introduce a practical network architecture which reduces link oversubscription by using flat addressing, Valiant Load Balancing, and end-system based addressing, while attempting to allow dynamic resource allocation in large EDCs[16]. Al-Fares et al. present a system design with which they argue that by properly architecting and connecting commodity ethernet switches and computer clusters, a system may be able to deliver more performance at a lower cost than available from today's higher-end solutions[1]. Most of the available literature focuses on solving QoS management, and performing resource protection by proposing system architectures, usually involving over provisioning, or which perform dynamic allocation of system resources.

2.5.2. Existing Solutions for Service Traffic Shaping

We have so far identified in the literature two different strategies for performing service traffic shaping in service-oriented EDCs. The simplest strategy is to use a Manual and Static Allocation (MSA), in which the allowed rate is equally divided among all the SON appliances:

$$x_i = \left\lfloor \frac{X \times T}{B} \right\rfloor, \quad \forall i \in [1, \ldots, B], \tag{2.1}$$

where x_i is the number of credits allocated to appliance i, X is the maximum rate allowed per enforcement period, T is the duration of the enforcement period, and B is the number of appliances. This solution, although simple, is quite inefficient as it only provides satisfactory performance when the incoming traffic rates at the appliances are identical. Therefore, a number of appliances may hold queued requests while others remain idle.

CASTS[6], is a solution that relies on the communication and processing capabilities of the appliances in order to provide a better response to the requirements

specified in the CSC. In summary, CASTS works as follows. It proposes dividing the SAR enforcing period into *K subperiods* (see Fig. 2.6), during which the traffic is measured and the rate adapted also by means of assigning credits to the appliances. During subperiod k, each appliance estimates the number of credits it will have during the next interval using queue sizes, measures the number of requests queued and already sent, and broadcasts these values to the other appliances. Each appliance updates its shaping rate for subperiod $k+1$ after receiving the information from all the other appliances as follows:

$$x_i(k+1) = \left\lfloor D \times \frac{Q_i(k)}{\sum_{j=1}^{B} Q_j(k)} \right\rfloor, \tag{2.2}$$

where $Q_i(k)$ is the number of queued requests at appliance i, D is the number of remaining credits of the enforcement period, and B is the number of appliances. Note that an approximation function (in this case, a *flooring* function) is necessary, as the CSC specifies an *integer* number of requests to be sent to the service tier. This solution guarantees that the SAR is respected at all times by assigning credits dynamically under a weighted strategy. This approach uses MSA to adjust the number of credits during the first subperiod and when there are no queued requests in any appliance.

2.5.3. Rationale for contributions

In order to comply with the CSC, the calculations used in existing credit-based approaches lead to three main drawbacks which will be discussed in more detail in Chapter 3: the *flooring* effect which wastes systems resources, *starvation* or unfair credit allocation, and *fast start* or rapid credit consumption. Moreover, existing approaches only consider the *multipoint-to-point* case, and therefore no service traffic shaping solutions exist for the *multipoint-to-multipoint* case. Finally, as current approaches are based on the assumption that all entities are located in the same geographical location, they are inapplicable for geographically distributed EDCs, as they do not take into account communication delays.

2.6. Conclusion

In this chapter, we discussed the problem of resource protection in two-tier Enterprise Data Centers (EDCs). In many modern EDCs, to relieve service servers from the computational cost of CPU intensive tasks like XML parsing and to perform

resource protection, these functions are offloaded on a cluster of SON appliances on the preprocessing tier. Currently, SON appliances are limited in the sense they are only able to protect EDC resources by limiting the *number of requests per second* a server receives or, in other words, by *shaping* the service traffic. We argue that, because of the particularities of two-tier EDCs, classical traffic engineering shaping solutions cannot be used. Moreover, the few current approaches found in the literature, which are specifically designed for two-tier setups, are limited to EDCs on a single-site offering a single-service, and present flaws which render them inefficient.

Chapter 3

DoWSS: Multipoint-to-Point Resource Protection

In this chapter, we specifically consider the service *traffic shaping* problem where several access points (i.e., SON appliances) access concurrently a single service host (i.e., multipoint-to-point case). The typical solution consists of using a manual and static allocation strategy, in which the allowed rate is equally divided among all the access points. This solution, although simple, is quite inefficient as it only provides satisfactory performance when the incoming traffic rates at the SON appliances are identical. In[6], the authors proposed a better, more dynamic solution that monitors the traffic on a regular basis and adapts the rate by reassigning credits to each appliance under a weighted strategy based on queue sizes. In order to comply with the SAR, the calculations used in existing credit-based approaches lead to three main drawbacks:

1. *Flooring effect.* Existing credit-based solutions require the use of a flooring function to approximate the results to the integer immediately below. In some cases, when the number of appliances is not a divisor of the available credits, the use of a flooring function leads to under-utilization of the system.

2. *Fast start.* When the system operates under high input rates, all the available credits are rapidly consumed early in the enforcement period. This may result in overwhelming the service host, since a large number of requests are being sent during a time period substantially smaller than the specified enforcement period.

3. *Starvation.* The weighted strategies used for dynamic credit allocation are based on queue sizes. As a consequence, the appliances with at least one

queued event may be allocated all the credits, thus depriving the appliances with empty queues from credits[1].

The immediate repercussion of these issues is that they lead to suboptimal performance, as ti will be demonstrated later on this chapter. *Given the costs of implementing SON and issues inherent to the provision of Web Services, it is imperative to design efficient algorithms that optimize the overall utilization of the system.*

In order to solve the issues cited above, we propose DOWSS (**Do**ubly-**W**eighted algorithm for **S**ervice traffic **S**haping), an algorithm for service traffic shaping. Our approach is based on dividing the enforcement period into several smaller *enforcement subperiods* and on calculating the maximum allowed rates avoiding the use of flooring functions. By using a doubly-weighted strategy, our approach prevents starvation issues that may appear under certain conditions. We also introduce a procedure to contain the rapid consumption of credits at the beginning of the enforcement period, when there are high input rates to the system. Through simulation analysis, we show that our approach not only outperforms existing approaches, but it also has a substantial positive impact on the overall performance of the system over time.

In summary, the contributions of this work are:

- We identify three issues present in existing credit-based service traffic shaping approaches: *flooring*, *starvation* and *fast start*.
- We propose a dynamic, doubly-weighted, credit-based approach, that avoids the *flooring* and *starvation* issues, and uses system resources to their fullest.
- We introduce a contention mechanism to minimize the *fast start* phenomenon, which manifests itself when the input rate to the system is much greater than the Service Access Requirement rate.

The remainder of this chapter is structured as follows. In Section 3.1, we introduce the specific system architecture we focus on. Related work specific to the multipoint-to-point case is presented in Section 3.2. In Section 3.3, we perform a detailed analysis of existing issues in multipoint-to-point service traffic shaping in SON. In Section 3.4, we introduce our approach, while in Section 3.5 we evaluate our algorithm via extensive simulations. We discuss our results in Section 3.6. Finally, we conclude this chapter in Section 3.7.

[1]Note that the definition of starvation used throughout this chapter differs from that used in scheduling literature, where a process is perpetually denied necessary resources, and therefore can never finish its task[33].

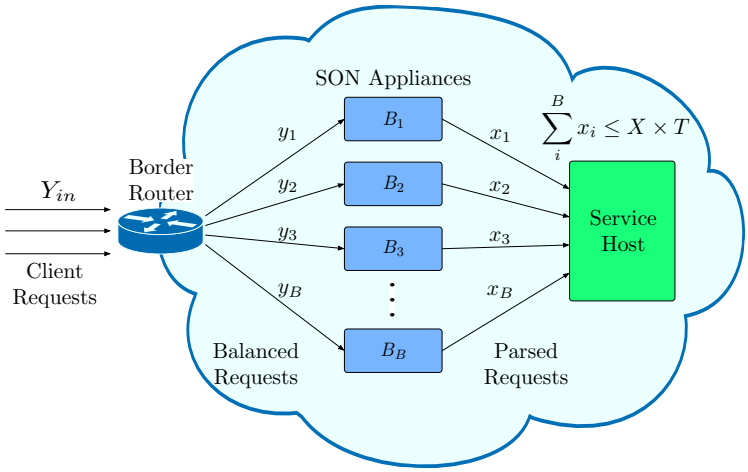


Figure 3.1: System Architecture showing the different elements and parameters involved in the service contract enforcement procedure for the *multipoint-to-point* case.

3.1. Instantiation of the System Architecture to the Multipoint-to-Point Case

The general architecture of the considered system is depicted in Fig. 3.1. There are four main elements composing this architecture:

- **Clients:** The clients are nodes that generate service requests. They can be located anywhere in the Internet. In Fig. 3.1, Y_{in} denotes the input rate of requests into the system.
- **Gateways:** These are border routers. They are responsible of forwarding service requests to the SON appliances and of distributing the service load among the appliances without any deep-content inspection.
- **SON Appliances:** These are middleware devices responsible for translating XML requests into the system's local language. They are also responsible for controlling the rate at which the service requests are forwarded to the service host. In the figure, x_i denotes the number of processed requests appliance B_i sends to the service host within some time interval.
- **Service host:** This node handles processing of the requests. It also specifies the rate at which the services may be accessed or SAR. In the figure, the service may not receive more than C requests during an time interval of duration T, where $C = X \times T$.

As mentioned earlier in this document, in SON environments, clients may access services from multiple entry points. In this chapter, we specifically consider the service traffic shaping problem where several access points (i.e., SON appliances) access concurrently a single service host (i.e., multipoint-to-point case). Even though approaches dealing specifically with this kind of setups exist in the literature, they present a number of drawbacks which prevents them from efficiently performing multipoint-to-point service traffic shaping. We formalize and analyze these issues in this the following section.

3.2. Related Work Specific to the Multipoint-to-Point Case

Many research efforts in the literature are oriented specifically towards QoS control in Web server farms. Nevertheless, these efforts center around enforcing SLAs defined in terms of maximum response times of a service. To this end, most of the methods are based on either service differentiation and prioritization[30;44], or admission control of new service requests[8;9], or both[3;20].

In particular, the work that is the closest to ours is the one of Garcia et al.[14]. The authors analyze the deficiencies in existing methods, define a number of requirements that any QoS control mechanism working in an enterprise environment should address, and establish their basic design principles. Along these lines, the authors propose a QoS control mechanism which determines the maximum number of concurrent sessions that a service host can process, and calculates the maximum admissible sessions in order to maintain the values specified by the SLA.

It is worth noting that, most of these research works involve the use of a centralized server and measurement/processing at the service hosts. Since the idea behind using a multi-tier architecture is to offload some tasks to be done from the servers, these methods are not applicable in our particular context. In contrast, our work aims at a decentralized method for enforcing the SAR. Furthermore, we do not center our work on service response times or client differentiation and scheduling. Instead, our objective is to prevent service hosts from being overwhelmed.

3.3. Issues with Existing Multipoint-to-Point Solutions

In this section, we identify and address three problems associated with existing service traffic shaping solutions. We start by defining them in the following.

3.3.1. Flooring Effect

SAR are specified in terms of requests per time unit within the observation period. Consequently, in order to comply with the SAR, the calculations conducted in both solutions involve the use of a *flooring function* to approximate the number of allocated credits to the integer immediately below. In some cases, when the number of appliances is not a divisor of the number of available credits, the use of a flooring function leads to under-utilization of the system. Fig. 3.2 depicts a sample of the typical performance of CASTS and MSA for different input rates (Y_{in}), with an enforcement period (T) of 1 second, using ten SON Appliances and a maximum allowed rate (X) of 128 requests per second represented by a horizontal dotted line. Even though both approaches process a number of requests near $C = X \times T = 128$, they never reach the maximum value. Therefore, at each enforcement period, there are a number of requests that are left unprocessed and accumulate significantly over time (see Fig. 3.6). As a consequence, the system is unable to exploit its maximum capacity. *Given the costs of implementing SON and issues inherent to the provision of Web Services, it is imperative to design efficient algorithms that optimize the overall utilization of the system.* Achieving optimal performance is fundamental in the long term.

Definition 1. (*Optimality*) Let $R(K)$ be the total number of processed requests within the observation period T, $X \times T$ be the maximum allowed number of requests to be sent to a service host during an observation period T, and $Y_{in} \times T$ be the total number of requests generated within an observation period T. We say that a shaping algorithm is *optimal* if $R(K) = \min[X \times T; Y_{in} \times T]$.

3.3.2. Starvation

Even though the gateway performs some sort of load balancing, this does not guarantee that the load will be equally distributed among all appliances. Indeed, as requests require different processing times, load balancing at the gateway is not transferred to the output rates of the SON appliances. As a consequence, since both MSA and CASTS are based on the allocation of credits that the appliances will use to send requests to the service hosts, a starvation phenomenon appears, in which some

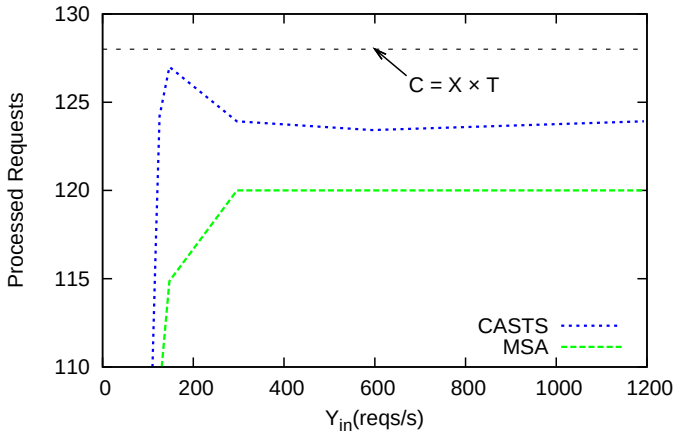


Figure 3.2: Performance of CASTS and MSA for different input rates and $X = 128$ reqs/sec.

appliances will have available credits, while others remain idle. This phenomenon is evidenced in two different ways, depending on the credit allocation scheme used.

When using MSA, an appliance is allocated a number of fixed credits, and it can use them to send requests to the service host as fast as it can. Furthermore, since this method does not involve any kind of communication between appliances, an appliance is not able to know if the other appliances have queued requests. Consequently, when it operates under high (instantaneous) input rates, an appliance receiving more requests will rapidly consume all of its credits, while other appliances receiving a smaller amount of requests will have more credits available even though they will not use them.

CASTS relies on communication between appliances, and each appliance informs the others of the number of credits it has consumed so far and the number of request that it has still queued. The appliances use this information to perform a weighted assignment of credits, in which the appliances with the largest queues will get the most credits. Nevertheless, this scheme penalizes the appliances with little or no queued requests. The weighted allocated scheme used in CASTS attempts to estimate the sending rate to the service host during the following interval using queue sizes. However, it cannot predict how many requests will enter the system during the next interval, and if these new requests will be equally balanced among all appliances. Therefore, it is possible that an appliance with a larger queue and more credits will continue to accumulate requests without being able to process them; at the same time, an appliance with a small queue that was allocated a few credits will consume these few credits before receiving new requests, remaining idle during the entire subperiod even though it is capable of processing additional requests.

3.3.3. Fast Start

An observed phenomenon in both approaches is the high sending rate of requests to the service host during a small period of time at the beginning of each enforcement period. We refer to this phenomenon as *fast start*. MSA and CASTS use different methods for allocating credits to the appliances. Nevertheless, the allocation does not specify *how fast* the credits should be used. Therefore, when there are high input rates, the appliances will rapidly consume all their credits during the early moments of the enforcement period by processing and sending requests to the service hosts as fast as possible. Even though the idea behind the SAR is to prevent the service hosts from being overwhelmed, sending a large number of requests during a time period substantially smaller than the specified enforcement period could result in effectively overwhelming the service host.

This type of behavior is also associated with the ill-defined nature of the SARs in SON. In the ATM network environment, for example, the SARs include the SAR definitions for the Maximum Burst Size (MBS) and a Cell Delay Variation Tolerance (CDVT). In this way, the number of cells per time period arriving at a host is easily limited. Since in a SON environment the SARs do not incorporate equivalent definitions of burst sizes and maximum delays, undesirable effects such as the *fast start* appear, resulting in a substantial impact on the overall efficiency of the system.

It is worth noting that, even though *fast start* is an undesired phenomenon, it is nonetheless inherent to the way these algorithms work. Indeed, since both allocation methods are based on distributing all available credits among all SON appliances, the *fast start* will depend directly on the maximum allowed rate to enforce and the number of SON appliances used. For instance, suppose that we have $B = 32$ SON appliances and that the SAR rate specified is $X = 128$ reqs/sec. When the instantaneous input rate is much larger than the SAR rate ($Y_{in} \gg X$, e.g., $Y_{in} = 3,000$ reqs/sec), even if each SON appliance is limited to forward only one request per enforcement subperiod, the entire system will nevertheless send 32 requests to the service host during the first enforcement subperiod. Therefore, the service host will receive 25% of the maximum allowed rate during the first subperiod alone.

3.3.4. Discussion

The immediate consequence of these issues is that they lead to "suboptimal performance". The main issue present in existing credit-based approaches is the use of flooring functions to approximate the number of allocated credits to the integer

immediately below. In order to avoid this issue, another kind of approximation function should be used. The *starvation* issue is mainly due to the use of a weighted strategy based only on the number of queued requests. This issue could be avoided by taking a more thorough look on the characteristics of the queued requests. Since the *fast start* issue is inherent to the way credit-based algorithms work, it is not possible to completely avoid it. Nevertheless, a mechanism that partially contains the effects induced by the *fast start* could be designed and incorporated into the overall management system.

3.4. DoWSS: Doubly-Weighted algorithm for Service Traffic Shaping

To address the existing issues in SAR enforcement in Service-Oriented Networks, we propose DoWSS, a doubly-weighted algorithm for service traffic shaping in service-oriented networks. Our approach is especially designed for Web Server farms implementing the multipoint-to-point access strategy. In this section, we first specify the preliminaries of our approach. The details of DoWSS come thereafter.

3.4.1. Preliminaries

DoWSS is based on the notion of *enforcement subperiod* introduced in CASTS[6]. The enforcement period is divided into K subperiods (see Fig. 2.6). During each subperiod, the algorithm will measure the number of requests that were processed and forwarded to the service host and queue sizes, and adapt its sending rate for the next subperiod by assigning *credits* to each appliance. A *credit* allows an appliance to send a processed request to the service host.

SARs are specified in requests per second within the enforcement period. Consequently, credit-based approaches must approximate the number of allocated credits by an integer value. The main difference between DoWSS and existing credit-based approaches is the type of function used to approximate the number of allocated credits to the integer. Instead of using a flooring function, like other approaches, DoWSS uses a ceiling function. Even though the use of a ceiling function may lead to a non-compliance of the maximum request rate, the SAR can still be enforced by using the communication capacities of the appliances. Since the SAR is a fixed value, all SON Appliances have this information beforehand. Therefore, when calculating the rate allocated to each appliance at each subperiod, the non-compliance

of the SAR can be detected. If the credits that are to be allocated to each appliance exceed the number of remaining credits, one or several appliances will be *penalized* by having credits reduced, so that the SAR is respected. This penalization is done randomly, and the procedure varies depending on whether or not there are queued requests in any appliance.

3.4.2. Case I: Empty Queues

First, the number of credits/requests available for allocation is calculated:

$$D = X \times T - R(k-1), \tag{3.1}$$

where $R(k-1)$ is the number of requests processed up until the latest subinterval. By definition, $R(0) = 0$. At the beginning of the first enforcement subperiod, and in subperiods in which there are no queued requests in any appliance, MSA is used for credit allocation:

$$x_i(1) = \left\lceil \frac{D}{B} \right\rceil, \quad i = 1, \ldots, B. \tag{3.2}$$

Once each appliance has calculated the number of credits allocated to it, we compute P, which is the number of credits exceeding the SAR:

$$P = \left(\sum_{n=1}^{B} x_n(k) \right) - D. \tag{3.3}$$

If $P > 0$, appliance B_i will generate then a random number N_i. This number is broadcast to the rest of the appliances. When all the appliances finish this information exchange, each appliance i will find the P lowest numbers among the numbers generated by all the appliances. If its own randomly generated number N_i is among the P lowest, the appliance has one of its credits removed. Otherwise, the appliance keeps all of its assigned credits. Note that this exchange of random values can be done both in a centralized or distributed manner. By design choice, we opt for the distributed way.

To reduce the possibility of conflicts between appliances (i.e., two or more appliances generating the same number), N should be chosen in a range much larger than the number of appliances. Nevertheless, it is still possible for two appliances to generate the same random number (see details in Section 3.4.4).

3.4.3. Case II: Non-empty Queues

When there is at least one queued request in the entire system, the use of the weighted strategy proposed in[6] may lead to *starvation* in appliances with empty queues (cf., Section 3.3). To avoid this, a doubly-weighted strategy is proposed. First, the n-th request in the queue is assigned a weight w_n, calculated as

$$w_n = \log_{10}\left(\frac{V}{s_n}\right), n = 1, \cdots, Q, s_n \neq 0, \tag{3.4}$$

where V is the processing speed of the appliance in bits per second and s_n the size of the n-th request, measured in bits. For simplicity, we make the assumption that, on average, the processing time of a request is proportional to the length (size) of the request[2]. The weight of a request is therefore inversely proportional to its size and depends directly on the number of measurement subperiods used. Therefore, large requests, which take longer to process, will have smaller weights. When the request processing time is larger than T/K, the weight of the request is negative. In this case, the weight of the request is set to zero. If an appliance has an empty queue, the weight is calculated using a virtual file size of 1 bit. We use a logarithmic scale in our calculations in order to work with numbers in a smaller range. The weight of appliance B_i is the sum of the weights of all the requests in the queue:

$$W_{B_i} = \sum_{n=1}^{Q_i} w_n. \tag{3.5}$$

Once each appliance calculates its own weight, it calculates the number of credits it is allocated during the next subperiod under a weighted strategy:

$$x_i(k) = \left\lceil D \times \frac{W_{B_i}(k)}{\sum_{n=1}^{B} W_n(k)} \right\rceil \cdot \tag{3.6}$$

Since the number of allocated credits is calculated locally, each appliance broadcasts this value to the other appliances. Upon reception of the information coming from other appliances, each appliance calculates the number of exceeding credits P using Equation 3.3.

Then, in order to specify which appliances are to be penalized, the same procedure used with empty queues is used. Each appliance will generate a random number N. This number is broadcast to the other appliances. Once an appliance

[2]In reality, the average processing time is proportional to length of the requests (e.g., due to parsing the entire XML document for checking well-formedness) as well as other factors, like the actual content of the XML document.

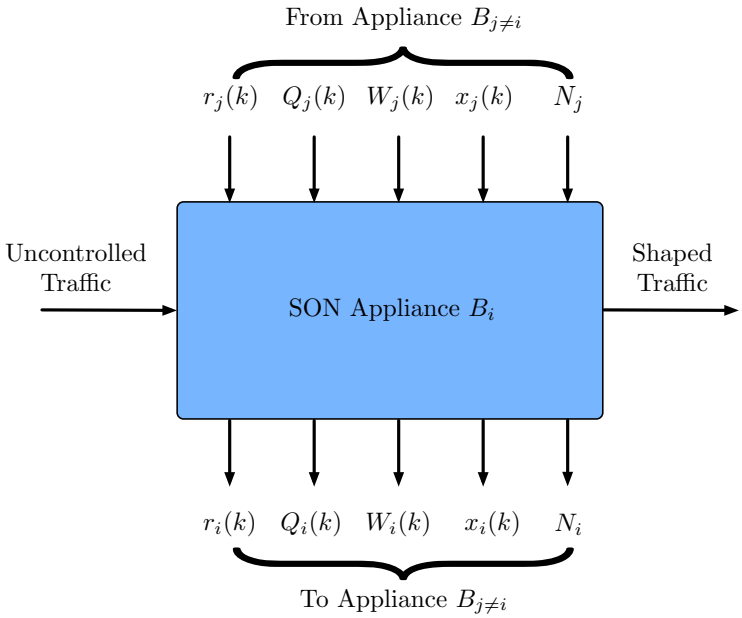


Figure 3.3: DoWSS information exchange during the k-th enforcement subperiod.

receives all the information coming from other appliances, it is penalized if its own randomly generated number N_i is among the lowest P. Fig. 3.3 depicts the information exchange during enforcement subperiod k. The entire procedure is summarized in Algorithm 1.

3.4.4. Conflict Resolution

To reduce the possibility of conflicts between appliances at credit removal, as explained before, random numbers are selected in a range that is much larger than the number of appliances. Nevertheless, it is still possible that two or more appliances generate the same random number. To avoid conflicts in this situation, the following procedure is followed. If all the "conflicting" numbers generated are among the P lowest random generated numbers, all involved appliances have their exceeding credits removed.

However, the conflicting generated numbers might be the greatest of the P lowest random generated numbers. For example, suppose that $B = 3$ is the number of appliances, $P = 1$ is the number of exceeding credits and $N = [100, 264, 264]$ is the set of random numbers generated by appliances 0, 1 and 2 respectively. In this particular case, the number of randomly generated numbers to penalize will exceed P, since appliances 1 and 2 generated the same number and it is greater than the number generated by appliance 0.

Algorithm 1 DoWSS algorithm.

Input: $k, N_j, \forall j \in [1, \ldots, B], j \neq i$.
Input: (When $1 < k \leq K$) $r_j(k-1)$, $Q_j(k-1)$, $W_j(k-1)$, and $x_n(k)$, $\forall j \in [1, \ldots, B], j \neq i$.
Output: new count $x_i(k)$.

if $k = 1$ **then**
  $x_i \leftarrow \lceil \frac{X \times T}{B} \rceil$
  $P = (x_i \times B) - (X \times T)$
  $N = \{N_1, \ldots, N_B\}$
  if $P \neq 0$ **then**
    Find P lowest of the set N
    if N_i is among P lowest **then**
      $x_i(k) \leftarrow x_i(k) - 1$
    end if
  else
    $x_i(k) \leftarrow x_i(k)$
  end if
else
  $r(k-1) = \sum_{j=1}^{B} r_j(k-1)$
  $R(k-1) = \sum_{n=1}^{k-1} r(n)$
  $W(k-1) = \{W_1, \ldots, W_B\}$
  $N = \{N_1, \ldots, N_B\}$
  if $R^j(k) < X^j$ **then**
    $D \leftarrow X \times T - R(k-1)$
    if $\sum_{n=1}^{B} Q_n(k) = 0$ **then**
      $x_i(k) \leftarrow \lceil \frac{D}{B} \rceil$
      $P = (x_i(k) \times B) - D$
      if $P \neq 0$ **then**
        Find P lowest of the set N
        if N_i is among lowest **then**
          $x_i(k) \leftarrow x_i(k) - 1$
        end if
      else
        $x_i(k) \leftarrow x_i(k)$
      end if
    else
      $x_i(k) \leftarrow \left\lceil D \times \frac{W_i(k)}{\sum_{n=1}^{B} W_n(k)} \right\rceil$
      $P = \left(\sum_{n=1}^{B} x_n\right) - D$
      if $P \neq 0$ **then**
        Find P lowest of the set N
        if N_i is among P lowest **then**
          $x_i(k) \leftarrow x_i(k) - 1$
        end if
      else
        $x_i(k) \leftarrow x_i(k)$
      end if
    end if
  else
    $x_i(k) \leftarrow 0$
  end if
end if

When this kind of situation occurs, a concurrency mechanism is used. When an appliance detects a value received from the other appliances equal to the value it has generated, it will notify the conflicting appliances of the equality and tell them to keep its assigned credit. To avoid the possibility of two appliances transmitting this notification at the same time and therefore causing further conflicts, this transmission is delayed by using a random timer. The first appliance whose timer expires, notifies the conflicting appliances, and will thus be effectively penalized.

3.4.5. Addressing the *fast start* issue

We introduce in our approach an optional method for avoiding the *fast start* issue (cf., Section 3.3.3). The idea is to further limit the number of requests sent to the service host during the beginning of each enforcement period, and distribute them over the entire observation period. Nevertheless, since this phenomenon only appears when the instantaneous entry rate exceeds X by a great value, the system will only activate the contention mechanism under these conditions.

At the beginning of each subperiod, the system can estimate the global input rate Y_{in} by measuring the number of received requests during the last subperiod. When a high input rate is detected, each appliance can calculate F, the "limited" sending rate towards the service hosts:

$$F = \left\lceil \frac{T \times X}{K \times B} \right\rceil . \tag{3.7}$$

Each appliance will then adjust its number of allocated credits to F for the subperiod, therefore $x_i(k) \leftarrow F$. Note that, with this optional method, by performing another manual rate assignment, the desired weighted effect is almost lost. If during the enforcement period the traffic abruptly changes from subperiod to subperiod (e.g., in the presence of bursty traffic), the contention method is turned off during the periods where the input rate is not greater than the SAR.

3.5. Evaluation

To study the performance of DoWSS, we undertook a series of simulations. To this end, the OMNeT++ Discrete Event Simulation System[35] was used. The OMNeT++ library controls the simulated time and the concurrent execution of the code running on each one of the simulated SON Appliances. All appliances run the same code. The algorithms are written in C++ and are event driven.

3.5.1. Experimental Setup

The simulation-based evaluation of DOWSS is centered around answering three questions:

1. Is the performance of DOWSS *optimal*?
2. Are the techniques used in DOWSS able to solve the *Flooring, Fast Start and Starvation issues?*
3. If input rates vary, will DOWSS be able to *adapt* to these variations while continuing to provide an *optimal performance*?

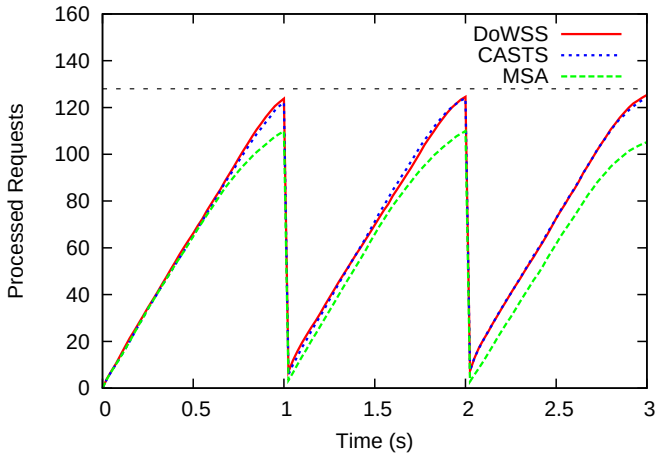
The first set of simulations aims to answer questions 1) and 2). For this set, the client service requests are modeled as Poisson processes. The average input rate to the system, noted as Y_{in}, is chosen as a fixed value (unknown, of course, to the SON Appliances) and is varied to verify SAR compliance for all input rates. The second set of simulations answers the 3rd question. To this end, we simulate bursty traffic using a Poisson Pareto Burst Process (PPBP) model[47]. Bursts are modeled as Poisson processes with a duration sampled from a Pareto distribution. In both sets, the processing rate of each document at each appliance varies and depends directly on document sizes. In a previous work[6], we explored the responsiveness of credit-based algorithms. We observed that for $T = 1$ and $K = 40$, the algorithm achieves a reasonable responsive behavior[3]. Therefore, for all the presented simulations, we have set $T = 1$ second, $K = 40$ and $X = 128$ requests per second, unless otherwise specified. All data points shown on the curves represent an average over 100 runs. We have calculated confidence intervals for each data point, however we do not show them on the curves for simplicity.

3.5.2. Performance Metrics

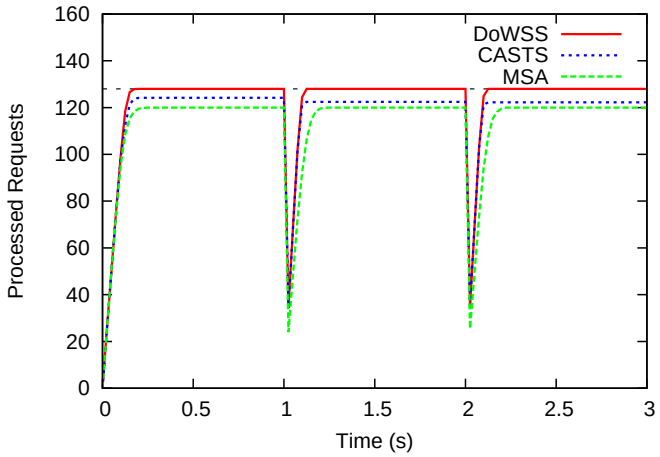
In order to properly measure the performance of DOWSS and compare it with other existing approaches, we use the following performance metrics:

1. *Processed Requests* (used in Fig. 3.4, 3.5, 3.8, 3.9, 3.10, and 3.11): Number of requests that were processed by the appliances and sent to the service host.

[3]In a real deployment scenario, the choice of the length of an enforcement period rests at the discretion of an IT administrator. The number of subintervals should then be chosen accordingly.



(a) $Y_{in} = 128$ reqs/s.



(b) $Y_{in} = 1000$ reqs/s.

Figure 3.4: Number of processed requests per enforcement period (T) for $Y_{in} = X$ and $Y_{in} >> X$ under uniform traffic

2. *Allocated Credits* and *Queued Requests* (used in Fig. 3.7): Respectively, the number of credits that were allocated to a particular appliance and the number of queued requests.

3. *Accumulated Unprocessed Requests* (used in Fig. 3.6): Number of unprocessed requests per enforcement period among all the appliances accumulated over time. Note that, as stated in Definition 1, the optimal algorithm should lead to $\min[X \times T; Y_{in} \times T]$ and therefore have zero unprocessed requests.

3.5.3. Performance under Uniform Traffic

In Fig. 3.4 the performance of DoWSS over one observation period is depicted, under the assumption of uniform traffic. This figure shows the number of processed requests during three enforcement periods. In Fig. 3.4(a), Y_{in} is set to 128 requests/s.

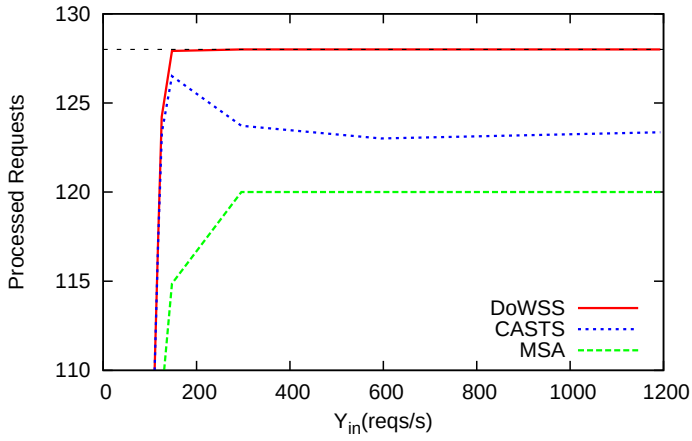


Figure 3.5: Total number of processed requests over one enforcement period for different input rates.

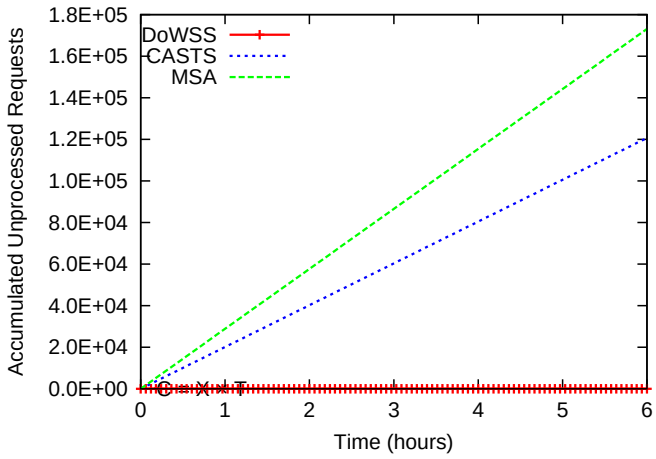


Figure 3.6: Total number of unprocessed requests over a period 6 hours for $Y_{in} = 300$ reqs/sec.

At the end of the observation period, using DoWSS and CASTS the system has processed more requests than MSA. Nevertheless, the value of processed requests never reaches $X \times T$, represented by a horizontal dotted line. This is due to the actual number of requests sent to the appliances. Since the data points represent averages over all the conducted simulations, in some cases Y_{in} might be less than $X \times T$. In Fig. 3.4(b), Y_{in} is set to 1,000 requests/s. In this case the contract limit X, represented by a horizontal dotted line, is achieved very early in the observation period. When using DoWSS the system performs optimally by processing exactly $X \times T$ credits.

In Fig. 3.5 we explore the performance of DoWSS during an enforcement period under uniform traffic. This figure shows the number of processed requests during one enforcement period, as a function of the input rate. The horizontal dotted line shows the value of $X \times T$. For entry rates much lower than X, both DoWSS

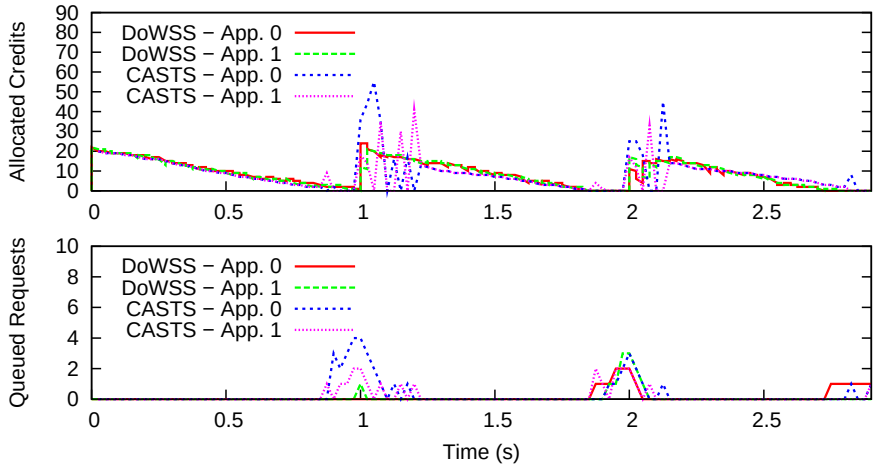


Figure 3.7: Number of allocated credits and queued requests on two appliances for $Y_{in} = 128$ reqs/s

and CASTS perform equally, as expected, while MSA shows a lower performance. However, for entry rates that are close to and above X, DoWSS outperforms both CASTS and MSA. Indeed, while CASTS obtains a good performance by processing a number of requests close to X, DoWSS performs optimally by processing and sending exactly X requests per observation period to the service host.

As mentioned before, the flooring function used in CASTS and MSA will limit in most cases the number of allocated credits to be less than $X \times T$, depending on the number of appliances used. Therefore, a number of requests are left unprocessed at the end of each enforcement period. *These requests accumulate over time, having a negative impact on the overall performance of the system.* Fig. 3.6 shows the impact of the flooring effect over time. After six hours running the three algorithms, DoWSS has processed up to 170,000 more requests than MSA, and around 120,000 more than CASTS. Clearly, by using DoWSS, the system exploits its maximum capacity.

To illustrate the starvation problem, Fig. 3.7 shows the number of allocated credits and queued requests for only two of the 10 used appliances. When using CASTS, during the first observation period, as there are no queued requests, the credits are uniformly distributed among all the appliances. During the subsequent observation periods, when there are queued requests, the appliance with the largest number of requests in queue will obtain most of the assigned credits. However, when there is only one appliance with queued requests, as is the case of appliance 1 around 1.25 seconds, it will claim all the credits during the subperiod. The other appliances are then left creditless, even with empty queues, and are therefore unable to process requests during the subperiod. DoWSS corrects this issue. After 1 second, appliance 1 has queued requests, while appliance 0 has an empty queue. Even with an empty

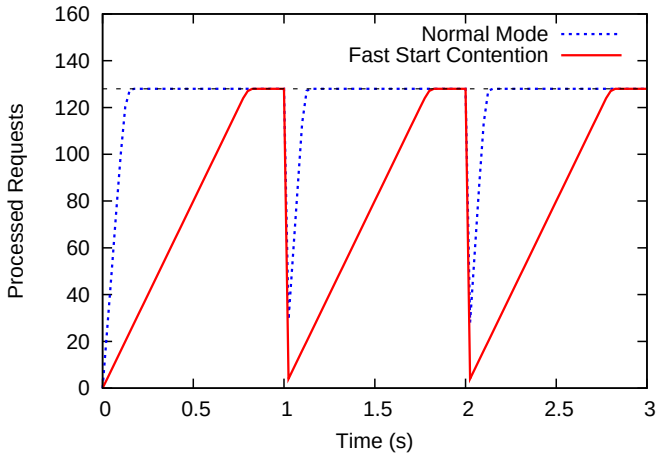


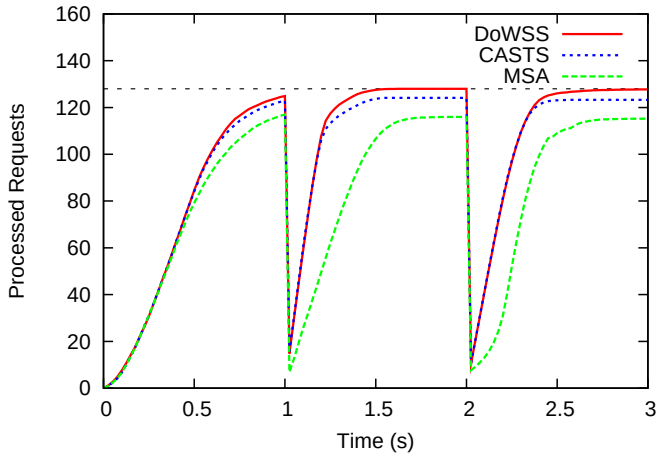
Figure 3.8: Performance of the fast start contention mechanism under uniform traffic.

queue, appliance 0 is allocated credits, and thus it is able to process any requests that arrive during the subperiod.

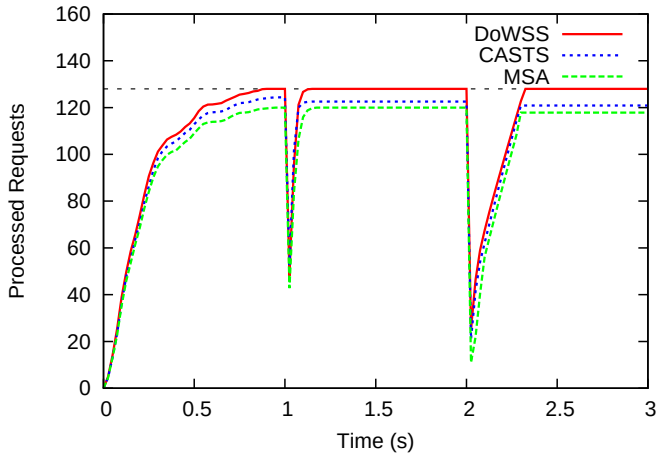
In Fig. 3.8, we illustrate the performance of the fast start contention mechanism under uniform traffic. Since the fast start depends directly on the number of SON Appliances used (cf. Section 3.3.3), for this set of simulations we set $B = 4$ and $Y_{in} = 1,000$ reqs/sec. If the fast start contention method is not used, the available credits are rapidly consumed during the first few enforcement subperiods, therefore sending the entire number of requests allowed by the SAR during a short period of time. This situation could result in effectively overwhelming the service host. When the contention method is used, the credits are consumed during the entire duration of the enforcement period. In this way, DoWSS further prevents the service host from being overwhelmed.

3.5.4. Performance Under Bursty Traffic

In Fig. 3.9, the typical performance of DoWSS under bursty traffic is observed. This figure shows the number of processed requests during three observation periods. In Fig. 3.9(a), Y_{in} is set to 128 requests/s. Unlike previous performed simulations with uniform entry traffic, under bursty traffic the value of allocated credits reaches $X \times T$, represented by a horizontal dotted line, rather early in the enforcement period. This is due to the actual entry rate to the system. Since the requests are sent in bursts, the number of requests sent to the SON appliances may reach the maximum specified value at any point in the enforcement period. At the end of the first observation period, DoWSS and CASTS have allocated more credits than MSA. In Fig. 3.9(b), Y_{in} is set to 1,000 requests/s. In this case, the performance of DoWSS is



(a) $Y_{in} = 128$ reqs/s.



(b) $Y_{in} = 1000$ reqs/s.

Figure 3.9: Number of processed requests per enforcement period (T) for $Y_{in} = X$ and $Y_{in} >> X$ under bursty traffic.

similar to the performance under uniform traffic. For both entry rates, the performance of CASTS and MSA deteriorates over time.

In Fig. 3.10, we explore the performance of DoWSS as a function of the input (bursty) traffic. This figure shows the number of processed requests during one enforcement period. The horizontal dotted line shows the value of $X \times T$. Even with bursty traffic, for entry rates much lower than X, both DoWSS and CASTS perform equally and as expected, while MSA shows a lower performance. However, for entry rates that are close to and above X, DoWSS outperforms both CASTS and MSA. Indeed, CASTS obtains a good performance by processing a number of requests close to X, for input values close to X. However, the performance of CASTS decreases as Y_{in} increases. On the other hand, DoWSS performs optimally by processing X requests per observation period.

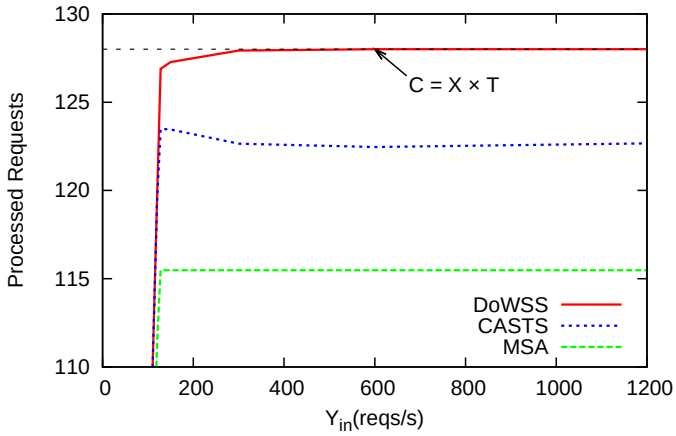


Figure 3.10: Total number of sent requests for different input rates under bursty traffic.

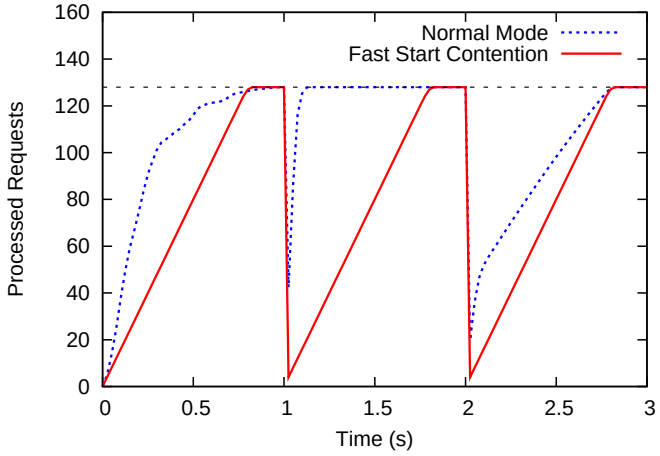


Figure 3.11: Performance of the fast start contention mechanism under bursty traffic.

In Fig. 3.11, we illustrate the performance of the fast start contention mechanism under bursty traffic. For this set of simulations, we set $B = 4$ and $Y_{in} = 1,000$ reqs/sec. If the fast start contention method is not used, after a short period of time without any received requests, the available credits are rapidly consumed during the few subsequent enforcement subperiods and therefore a substantial number of requests is sent to the service host during a small period of time. When the contention method is used, credit consumption is distributed almost homogeneously over the duration of the enforcement period, thus further preventing the service host from being overwhelmed.

3.6. Discussion

3.6.1. Optimality of Results

The obtained results show that the performance of both CASTS and MSA, although satisfactory, is below the performance of DoWSS. The difference is significant as it leads to increasing cumulative underperformance in the former cases. Indeed, since DoWSS complies with $R(K) = \min[X \times T; Y_{in} \times T]$, it has an optimal performance. The flooring function used in both existing credit-based approaches will limit in most cases the number of allocated credits to be less than $X \times T$, which is the maximum allowed by the SAR, depending on the number of appliances used. DoWSS will achieve this maximum while respecting the SAR. It is worth noting again that, even though the numerical difference between the results obtained using DoWSS and those obtained using CASTS is not large, the optimal performance of DoWSS has a substantial positive impact on the performance of the system over time.

A direct consequence of the suboptimal performance of existing credit-based approaches is the great amount of unprocessed requests which accumulates over time. Given the costs of implementing SON and issues inherent to the provision of Web Services, it is imperative to design efficient algorithms that optimize the overall utilization of the system. As DoWSS provides "optimal" performance, the system is able to exploit its maximum capacity. Therefore, the implementation of DoWSS maximizes the benefits of implementing a two-tiered architecture as the one considered in this thesis.

By avoiding the *starvation* issue, the algorithm prevents appliances from going idle in instances where their resources should be exploited. In other words, even with an empty queue, an appliance is allocated credits, and thus it is able to process any requests that arrive just after credit allocation has been performed. This further allows to use the system's resources to their maximum capacity.

The *fast start* contention method included in DoWSS service further enhances protection of the service host. Nevertheless, by using this optional method, another manual rate assignment is being performed, and therefore the desired weighted effect is almost lost. However, it is worth noting that, even though *fast start* is an undesired phenomenon, it is nonetheless inherent to the way credit-based algorithms work.

3.6.2. Communication Overhead

The proposed algorithm depends crucially on the information exchange between SON appliances in order to take decisions locally at each SON appliance. The amount of information exchanged during each enforcement period will depend directly on the number of enforcement subperiods defined. The goal of dividing the enforcement period into subperiods is to give the algorithm more chances to react to changes in the behavior of the input conditions. Nevertheless, the number of enforcement subperiods has a great impact on the behavior of the algorithm, since the higher the number of subintervals, the higher the control overhead.

The number of control messages exchanged is linearly proportional to both the number of appliances and the number of subintervals. Moreover, the duration of the enforcement period determines the optimum value of the number of enforcement subperiods. Indeed, a larger enforcement period could support a large number of subperiods. Consequently, depending on the length of the observation period and on the size of the requests, the control overhead might compromise the efficiency of the system.

3.7. Conclusions

In this chapter we presented DoWSS, a doubly-weighted algorithm for service traffic shaping in service-oriented networks. Contrary to existing credit-based approaches, our approach guarantees the allocation of at least one credit per measurement subperiod, thus effectively solving the numerical approximation issues, by exploring the communication capabilities of the SON Appliances. DoWSS involves the use of a doubly-weighted strategy for credit allocation, using weights based on request sizes. Therefore, DoWSS effectively penalizes the appliance queues that would take the longest to process, by assigning more credits to appliances with smaller queues, thus preventing *starvation*. The algorithm also introduces a procedure to contain the rapid consumption of credits at the beginning of enforcement period, or *fast start*, further preventing the service host from being overwhelmed.

We evaluated the performance of DoWSS by conducting a series of simulations. The obtained results show that DoWSS performs optimally by processing exactly $X \times T$ requests per observation period, which is the maximum possible number of requests allowed by the client service contract. We also show that our approach has a substantial positive impact over time on the overall performance of the system, by using it to its maximum capacity.

Chapter 4

MuST: Multipoint-to-Multipoint Resource Protection

Several approaches, using both static and dynamic credit-based strategies, have been developed in order to enforce the rate specified by the SAR[5;19]. Nevertheless, these solutions have so far only considered the *multipoint-to-point* case where a cluster of SON appliances *shapes* service traffic toward a *single* service instance. Moreover, current off-the-shelf SON Appliances present architectural limitations that prevent them from efficiently performing traffic shaping in the presence of *multiple* service hosts.

In this chapter, we identify the need for implementing multiple exit queues at each SON appliance when these are used to access multiple service instances. We propose MuST, an novel approach for *multipoint-to-multipoint* service traffic shaping. We show via simulation that our approach, which involves the use of a new SON appliance internal architecture combined with the strategic use of an efficient service traffic shaping algorithm, effectively solves the multipoint-to-multipoint service traffic shaping problem and pushes the system to its maximum capacity. In summary, the contributions of our work are:

- We identify the need for a queuing management scheme that is more adapted for scenarios where multiple appliances access concurrently multiple services.
- We propose the use of a novel internal SON appliance architecture, tied to an algorithm for shaping request traffic towards several services when the number of output queues is the same as the number of services.

- We validate our aproach via extensive simulations and show that MUST is able to push the system to its maximum capacity while respecting the service contracts.

The remainder of this chapter is structured as follows. We formalize the multipoint-to-multipoint service traffic shaping problem, and we clearly identify the shortcoming of current off-the-shelf SON appliances in Section 4.1. In Section 4.2, we present MUST, our approach for performing multipoint-to-multipoint service traffic shaping, before validating it through extensive simulations in Section 4.3. Related work on this area is presented in Section 4.4. Finally, in Section 4.5 we conclude this chapter.

4.1. Off-the-shelf SON Appliances and Traffic Shaping

In this section, we identify the existing practical issues for extending the multipoint-to-point case to the multipoint-to-multipoint case. We focus on the lack of an algorithm for efficiently performing service traffic shaping when in the presence of multiple services. We also investigate the qualitative shortcomings of off-the-shelf SON Appliances, namely their internal architecture, when used in this particular context.

4.1.1. From Multipoint-to-Point to Multipoint-to-Multipoint Shaping

In contrast to existing approaches, we specifically consider, in this chapter, the case where multiple SON appliances access concurrently multiple service hosts and must shape traffic towards the latter in order to protect them from being unduly overwhelmed. Fig. 4.1 depicts the considered system and parameters. We also consider that each different service host defines its own SAR and that all of the appliances in the cluster are able to process requests for all service instances.

We start by formalizing the per-service SAR. Let $x_i(s)$ be the number of requests appliance i is allowed to send to service host s (for the remainder of this chapter, we will refer to this value as i's credits for s). As per the SAR, the preprocessing tier must guarantee that the cumulated number of requests sent by all the appliances in the cluster towards service s must respect:

$$\sum_{i=1}^{B} x_i(s) \leq X(s) \times T. \tag{4.1}$$

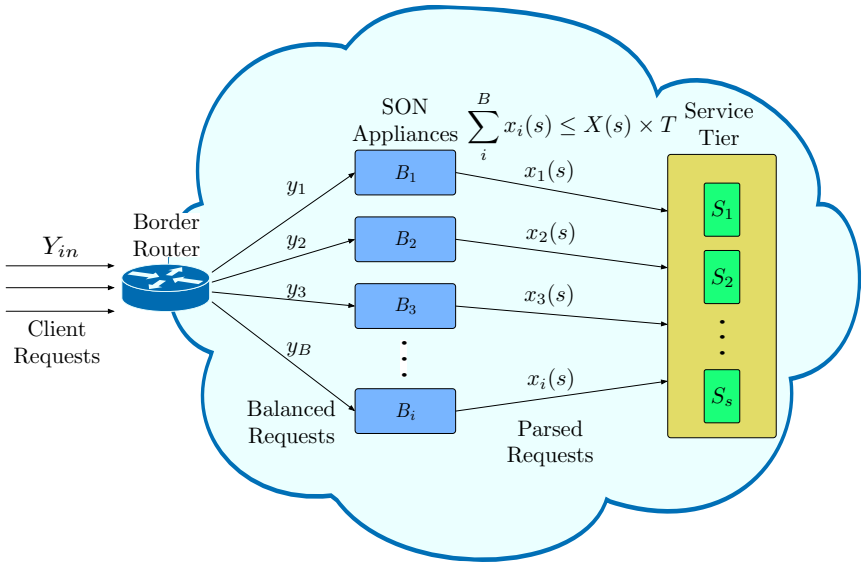


Figure 4.1: System Architecture showing the different elements and parameters involved in the service contract enforcement procedure for the *multipoint-to-multipoint* case.

To the best of our knowledge, there is no known published strategy for guaranteeing the specification described in Eq. 4.1. Consequently, in production environments, the simplest solution used nowadays is to apply the static, homogeneous policy or MSA in a way similar to the explanation given in Section 2.5.2. This policy assigns the same rate to each appliance at all times:

$$x_i(s) = \left\lfloor \frac{X(s) \times T}{B} \right\rfloor, \quad \forall i \in [1, \ldots, B]. \tag{4.2}$$

MSA, although simple, is quite inefficient as it only provides satisfactory performance when the incoming traffic rates at the appliances are identical. In practice, this is hardly the case as there is no a priori knowledge on the rates at which the preprocessing tier will receive requests from the clients. Moreover, even though the border routers perform some load balancing, since the delays required for request preprocessing are highly heterogeneous, the rate at which the appliances are ready to send documents to the service instances does not follow a uniform law. Therefore, a number of appliances may hold queued requests while others remain idle. As a consequence, the system is unable to exploit its maximum capacity as presented in Definition 1 below:

Definition 2. (*Optimality*) Let $R(s,T)$ be the total number of requests sent to service s within the observation period T, $X(s) \times T$ be the maximum allowed number of requests to be sent to service host s during an observation period T, and $Y_{in} \times T$ be

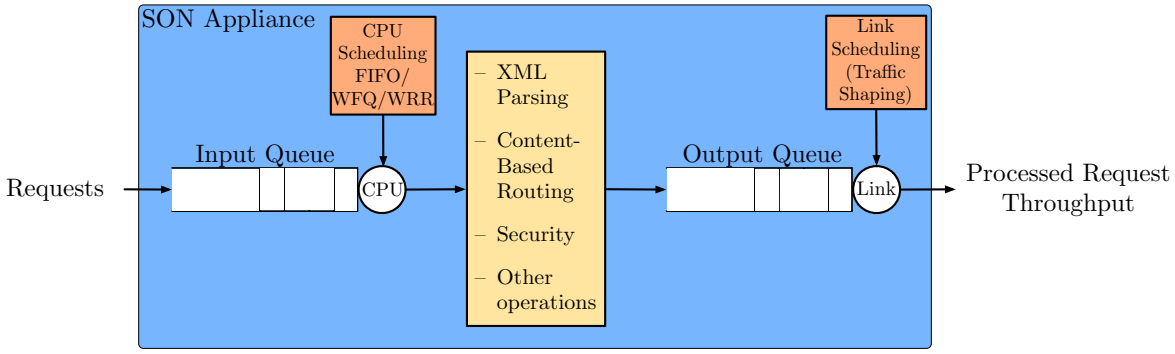


Figure 4.2: Internal architecture of off-the-shelf SON appliances.

the total number of requests generated within an observation period T. We say that a shaping algorithm is *optimal* if $R(s,T) = \min[X(s) \times T; Y_{in} \times T], \forall\, s$.

4.1.2. Architectural Shortcomings of Off-the-Shelf SON Appliances

The main building block of the preprocessing tier in the considered two-tier EDC is the set of off-the-shelf SON appliances. Fig. 4.2 shows the main internal components of a SON appliance. Inbound XML formatted requests are put in an entry queue where a CPU scheduler allocates the necessary resources for parsing requests and performing other operations as authentication and validation. Once a request is processed, it is placed in an output queue, which follows a FIFO service discipline, before being sent to the correct service host. At this stage, the appliance is responsible for enforcing each per-service SAR.

The current architectural design of off-the-shelf SON appliances makes them unfit for efficiently shaping traffic towards multiple different service hosts. Indeed, because of the use of a single FIFO output queue, as soon as the lowest per-service SAR is fulfilled, when a request for a service which no longer has credits reaches the front of the queue, it blocks all requests behind it even if there are credits left for other services. This shortcoming has major impact on the efficiency of the system, as it will be shown later in this section.

4.1.3. Arguments Towards New Algorithms

As mentioned before, the use of a single FIFO output queue severely limits the performance of SON appliances when shaping traffic towards multiple service hosts. To illustrate this, we undertook a series of simulations, where a cluster of six SON Appliances access concurrently a cluster of three service hosts. We define a differ-

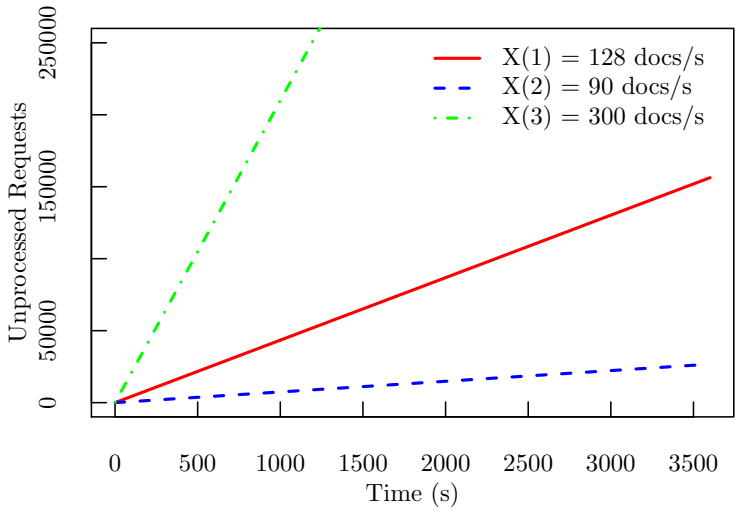


Figure 4.3: Number of requests left unprocessed overtime when using a static credit allocation scheme together with a single output FIFO queue for multipoint-to-multipoint service traffic shaping.

ent SAR for each service host. Fig. 4.3 shows the impact of using a single FIFO exit queue together with MSA over time. As per Definition 1, the optimal algorithm would leave no unprocessed requests overtime. Nevertheless, after only an hour of simulated time, over 150,000 requests for Service 1, around 3,000 for Service 2, and over 700,000 for Service 3 have been left unprocessed. Clearly, the qualitative shortcomings of both MSA and off-the-shelf appliances severely hampers the system. As a consequence, the system is unable to exploit its maximum capacity. *Given the costs of implementing EDCs and issues inherent to their provisioning, it is imperative to design efficient algorithms that optimize the overall utilization of the system.*

4.2. MUST: Multipoint-to-Multipoint Service Traffic Shaping

We first propose the requirements that SON appliances must fulfill in order to efficiently perform multipoint-to-multipoint service traffic shaping. We present then a credit-based algorithm designed for multipoint-to-multipoint traffic shaping which pushes the system to its maximum capacity.

4.2.1. A Novel SON Appliance Architecture

In order to properly perform service traffic shaping in two-tier EDC setups with multiple service hosts, we propose some simple architectural changes to current

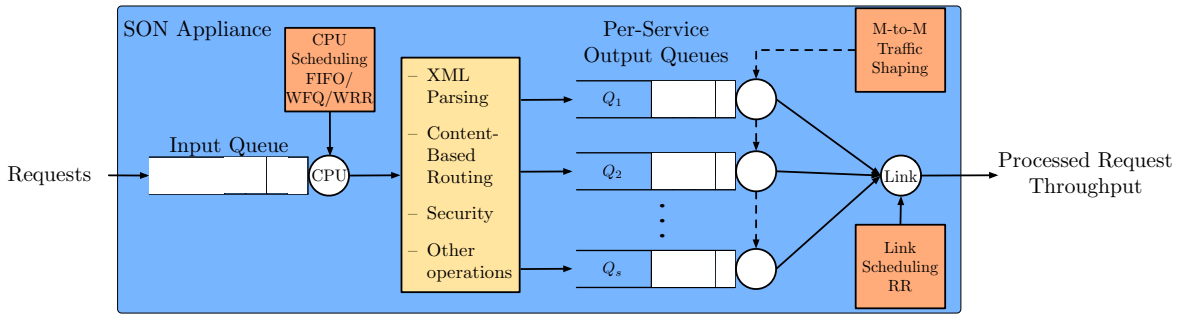


Figure 4.4: Proposed internal architecture of SON Appliances for multipoint-to-multipoint service traffic shaping.

off-the-shelf SON Appliances. First, we propose the use of a single output queue for each service present in the service tier. Second, because there are now several output queues accessing concurrently a single output link, we propose the use of a simple round-robin scheduling algorithm for sharing the link resource among the output queues. Fig. 4.4 depicts the proposed internal architecture.

4.2.2. A Multipoint-to-Multipoint Service Traffic Shaping Algorithm

As the next step, we propose the adaptation of DOWSS (cf. Chapter 3) in order to handle multiple services. First, the n-th request in the queue for service s is assigned a weight w_n. The weight of a request is inversely proportional to its size and depends directly on the number of measurement subperiods used. For simplicity, we make the assumption that, on average, the processing time of a request is proportional to the length (size) of the request. Therefore, large requests, which take longer to process, will have smaller weights. The weight of appliance B_i for service s, during subperiod k is the sum of the weights of all the requests in the output queue for service s:

$$W_{B_i}(s,k) = \sum_{n=1}^{Q} w_n. \tag{4.3}$$

Once each appliance calculates its own weight, it determines the number of per-service credits it is allocated during the next subperiod under a weighted strategy:

$$x_i(s,k) = \left\lceil D(s) \times \frac{W_{B_i}(s,k)}{\sum_{n=1}^{B} W_n(s,k)} \right\rceil, \tag{4.4}$$

where $D(s)$ is the number of preprocessed requests for service s, $W_{B_i}(s,k)$ is the weight of appliance i for service s, and $W_n(s,k)$ is the aggregate of the weights of all appliances in the cluster for service s. Note that an approximation function (in

this case, a *ceiling* function) is necessary, as the SAR specifies an *integer* number of documents to be sent to the service tier. By using ceiling function the maximum per-service allowed rate might be exceeded. In order to tackle this issue, appliances enter a "lottery" in which they exchange random generated numbers amongst them, and the appliances with the lowest numbers are "penalized" by having one of their credits taken away from them, depending on the number of credits that are exceeding the per-service SAR. This exchange of random values can be done both in a centralized or distributed manner. By design choice we opt for the distributed way. Moreover, to reduce the possibility of conflicts between appliances (i.e., two or more appliances generating the same number), random numbers should be chosen in a range much larger than the number of appliances in the cluster.

4.3. Evaluation

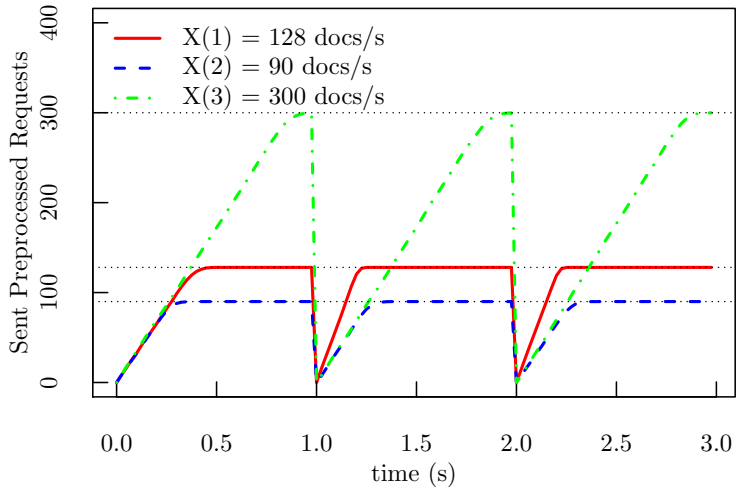
To study the performance of our multipoint-to-multipoint shaping approach, we undertook a series of simulations. To this end, we used the OMNeT++ Discrete Event Simulation System[35]. The OMNeT++ library controls the simulated time and the concurrent execution of the code running on each one of the simulated SON appliances. All appliances run the same code. The algorithms are written in C++ and are event driven. The simulation-based evaluation of our proposal is centered around answering three questions:

1. Is our approach able to solve the multipoint-to-multipoint traffic shaping problem?

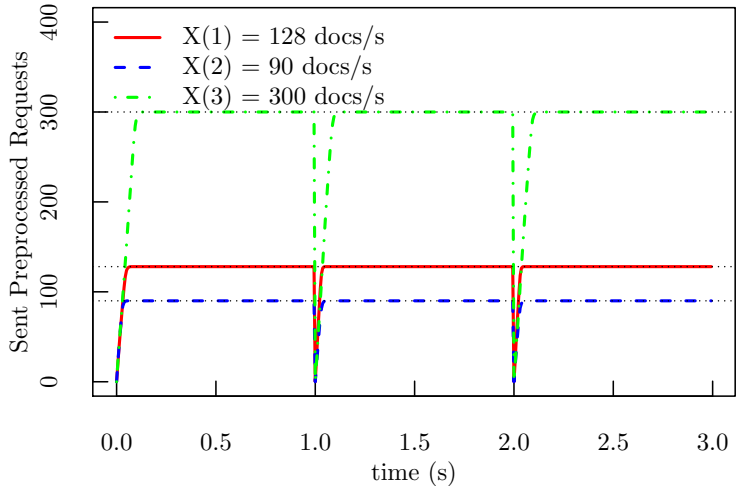
2. Are the techniques used by our approach better than the current techniques applied in production environments?

3. If input rates vary, will our algorithm be able to *adapt* to these variations while continuing to comply with per-service SARs?

To answer questions (1) and (2), we modeled client service requests as Poisson processes. The *average* input rate to the system, noted as Y_{in}, is chosen as a fixed value unknown to the SON Appliances; and is varied to verify SAR compliance for all input rates. We compared MUST with the solution used in production systems today (MSA and the internal off-the-shelf SON appliance architecture). Representative results are shown in Fig. 4.5 and 4.6 and are discussed later in this section. We proceed subsequently to answer question (3). To this end, we simulated bursty



(a) $Y_{in} = 1000$ reqs/s

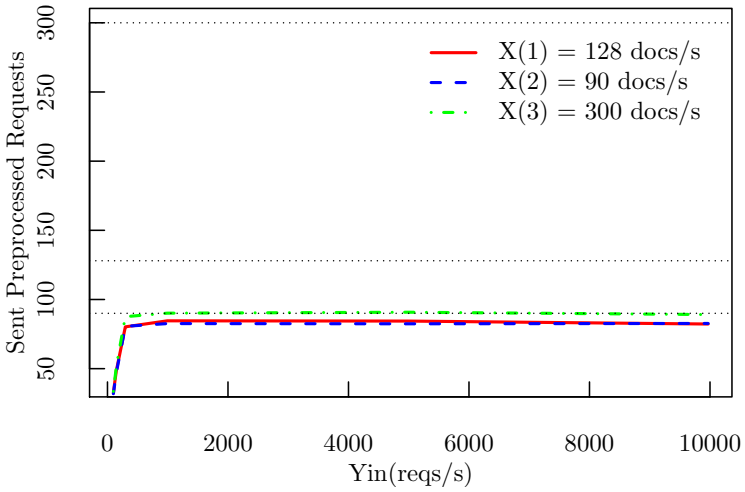


(b) $Y_{in} = 10000$ reqs/s

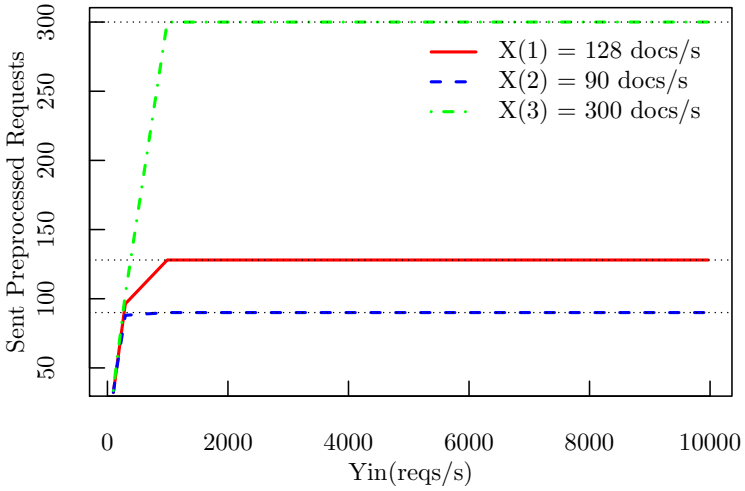
Figure 4.5: Performance of the algorithm for different input rates to the system during 3 enforcement subperiods.

traffic using a Poisson Pareto Burst Process (PPBP) model[47]. Burst arrivals are modeled as Poisson processes with a duration sampled from a Pareto distribution. Representative results are shown in Figs. 4.7 and 4.8.

For all simulations, we assume that all SON appliances are able to process requests for all service hosts. We also assume that the processing rate of each document at each appliance varies and depends directly on request sizes. Previous works have explored the responsiveness of credit-based algorithms[5;6]. Results show that for $T = 1$ and $K = 40$, the algorithm achieves a reasonable responsive behavior. Nevertheless, in a real deployment scenario, the choice of the length of an enforcement period rests at the discretion of an IT administrator. The number of subinter-



(a) MSA and current internal SON appliance architecture.

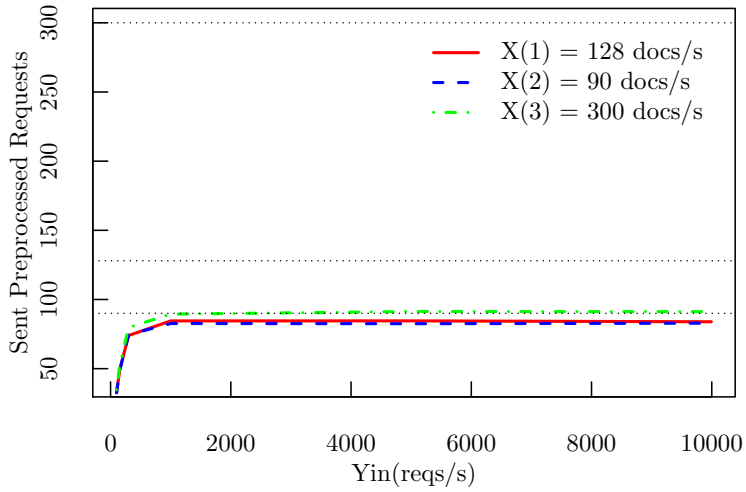


(b) MUST.

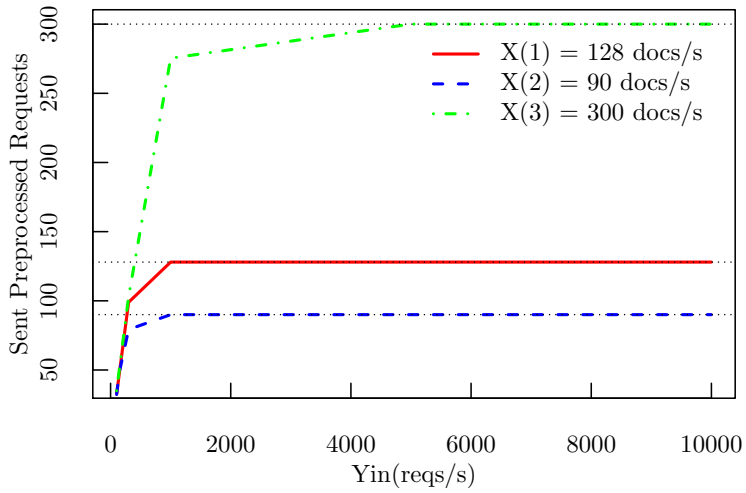
Figure 4.6: Comparison of the performance between a static allocation approach and our multipoint-to-multipoint approach for different input rates to the system.

vals should then be chosen accordingly. Therefore, for all the presented simulations, we have set $T = 1$ second, $K = 40$, $B = 6$, and $S = 3$. All data points shown on the curves represent an average over 50 runs.

In Fig. 4.5, the performance of our approach over three observation periods is depicted, under the assumption of uniform traffic. This figure shows the number of requests sent to each service host during three enforcement periods. In Fig. 4.5(a), Y_{in} is set to 1,000 requests/s. In this case, our algorithm is able to fully utilize the system's available resources sending exactly $X(s) \times T$ requests to the service hosts. Finally, in Fig. 4.5(b) Y_{in} is set to 10,000 requests/s. Even with a high input to the system, our algorithm is able to comply with the per-service SAR established by the



(a) MSA and current internal SON architecture.

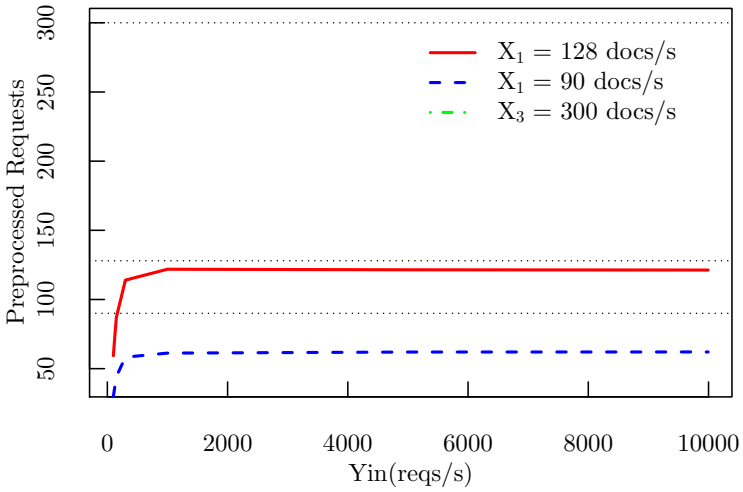


(b) MuST.

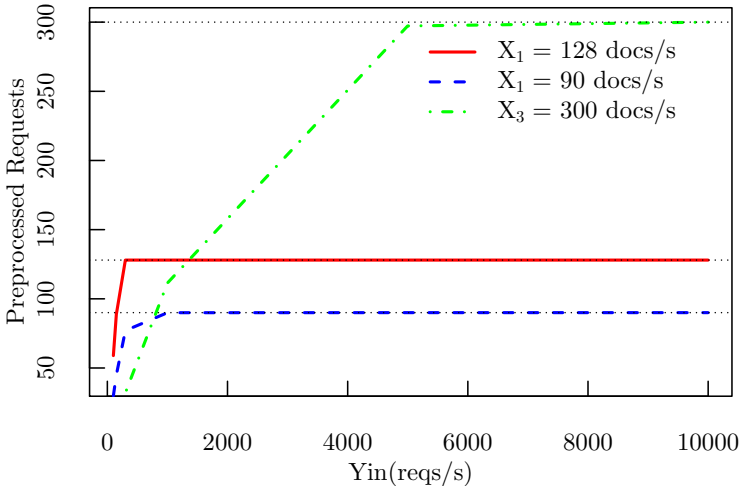
Figure 4.7: Comparison of the performance between a static allocation approach and our multipoint-to-multipoint approach for different input rates to the system under bursty traffic.

IT manager. Note that, in this figure the SAR is fulfilled very early in each observation period. This type of behavior is also associated with the ill-defined nature of the SARs as they do not specify *how fast* the credits should be used. However, even though this is an undesired phenomenon, it is nonetheless inherent to the way credit-based algorithms work.

In Fig. 4.6, we explore the performance of both our approach and MSA during an enforcement period under poisson traffic. This figure shows the number of requests sent to each service host during one enforcement period, as a function of the input rate. The horizontal dotted lines shows the values of $X(s) \times T$. For sending rates much lower than the lowest contract ($X(2) = 90$ requests/s), both schemes perform



(a) MSA and current internal SON architecture.



(b) Multipoint-to-multipoint approach.

Figure 4.8: Comparison of the performance between a static allocation approach and our multipoint-to-multipoint approach for different input rates to the system under bursty traffic when the traffic is unevenly distributed among services: (60% S(1), 30% S(2), 10% S(3)).

equally, as expected. However, for sending rates over $X(2)$, our algorithm outperforms MSA. Indeed, because of the use of a single FIFO output queue, as soon as the lowest SAR is fulfilled, when a request for a service which no longer has credits reaches the front of the queue, it blocks all requests behind it even if there are credits left for other services. On the other hand, our approach pushes the system to its maximum by processing and sending exactly $X(s)$ requests per observation period to each respective service host.

In Fig. 4.7, we explore the performance of our algorithm as a function of the input (bursty) traffic. The figure shows the number of requests sent to each service

host during one enforcement period, as a function of the bursty input rate. The horizontal dotted lines show the values of $X(s) \times T$. Even with bursty traffic, our algorithm is able to comply with the per-service SARs.

In Fig. 4.8 we explore the performance of both our approach and MSA during an enforcement period under bursty traffic. However, for this set of results the request traffic was unevenly distributed among the services as follows: 60% of the generated requests were bound towards Service 1, 30% towards Service 2, and 10% towards Service 3. The figure shows the number of requests sent to each service host during one enforcement period, as a function of the input rate. The horizontal dotted lines show the values of $X(s) \times T$. As expected, for sending rates much lower than the lowest contract ($X(2) = 90$ requests/s), both schemes perform equally. However, for sending rates over $X(2)$, we evidence a particular behavior. For MSA, the number of sent requests towards Service 1 closely approaches its contract ($X(1) = 128$ requests/s). Nevertheless, the traffic towards Services 2 and 3 is reduces dramatically in comparison to simulations where the traffic was distributed evenly among the services. Since more requests are being sent towards Service 1, the credits for this service will be rapidly consumed. In consequence, requests going to Services 1 and 2 will be blocked at the exit queue. On the other hand, we can evidence that our approach can dynamically adapt to this kind of scenarios. As a result contracts for all three services are completely fulfilled, thus pushing the system to be used to its maximum capacity. Note that, in the case of Service 3, the contract is attained around $Y_{in} = 5,000$ requests/s. This is due to the actual number of requests being sent towards Service 3.

4.4. Related work specific to the Multipoint-to-multipoint Case

Many academic and industrial research efforts found in the literature are oriented specifically towards the study and implementation of point-to-point shaping algorithms for the classic networking field. In this context, leaky bucket algorithms represent the state of the art[25].

For the multipoint-to-point shaping problem, the simplest strategy used today in production environments is to apply a static, homogeneous policy that divides the maximum allowed rate specified in the SAR between the SON appliances and the single service. This value is then used to assign the same rate to each appliance at all times. More recent efforts explore the communication capabilities of SON

appliances in order to a react to changes in the input conditions of the system and adapt output rates accordingly[5;19]. They also define algorithms where several virtual machines run on one host and share a single Ethernet link to send and receive traffic in a cluster or a data center[2].

It is worth noting that these solutions have so far only considered either the classic *point-to-point* or the *multipoint-to-point* traffic shaping case where a cluster of SON appliances *shapes* service traffic toward a *single* service instance. Moreover, as it was presented in Section 4.1, current off-the-shelf SON appliances present architectural limitations that prevents them from being used to efficiently perform traffic shaping in the presence of *multiple* service hosts.

4.5. Conclusion

In two-tier EDC setups, a cluster of SON appliances *locally* shapes the flow of client requests to enforce a *global* maximum access rate defined by a service host. In this chapter, we identified the architectural limitations present in off-the-shelf appliances in order to introduce a SO appliance architecture fit for *multi-service* traffic shaping. Subsequently, we proposed and validated via simulation MUST, a novel approach for *multipoint-to-multipoint* service traffic shaping in two-tier EDCs which solves the multipoint-to-multipoint service traffic shaping problem while pushing the system to its maximum capacity.

Note that, our approach relies on the use of a round robin service discipline in order to share the link resource among several exit queues. Even though other service disciplines can be used, in this chapter we focus on the core functionalities of our solution and deliberately do not conduct any study on the impact of other service disciplines on the performance of our approach. A more detailed study on this matter will be the subject of future work.

Chapter 5

GeoDS: Beyond Single-Site Enterprise Data Centers

Current solutions for service traffic shaping in service-oriented EDCs only consider the case where all the entities in it are located in a *single-site*. Nevertheless, as was previously stated, current trends point to having geographically-distributed EDCs. In this chapter, we focus on studying the service traffic shaping problem in *geographically-distributed* setups. We argue that, since network latencies are now an issue, shaping algorithms must be carefully designed as to include these new constraints. Indeed, the proposed shaping algorithms for both the multipoint-to-point and multipoint-to-multipont cases rely on the ability of appliances to communicate without accounting for network delays. However, because of network related delays, if there's a large network delay between an appliance an a service host, a request sent by an appliance during one enforcement period may arrive during the next enforcement subperiod, thus eventually leading to the non-compliance of the SAR.

To tackle the network delay-related issues inherent in geographical distribution we propose GeoDS, a geographically-distributed service traffic shaping algorithm. It relies on the knowledge of communication delays between all entities in the system. Based on this information, our approach makes credit allocation and request sending decisions aiming at reducing the effects of latencies no SAR compliance. In summary, the contributions of our work are:

- We identify the *late request arrival* issue in current shaping approaches which prevents them from using them in the presence of network latencies.

- We propose a geographically-distributed service traffic shaping algorithm that takes into account communication delays, thus preventing the *late arrival* of requests to the service hosts.
- We validate our approach via extensive simulations and we show that it is able to solve the service traffic shaping problem in the presence of non-negligible network latencies, while respecting the SAR.

The remainder of this chapter is structured as follows. In Section 5.1, we present the service traffic shaping problem in geographically-distributed setups, and identify the issues which arise when attempting to use current shaping solutions in this kind of deployments. In Section 5.2, we present GEODS, our approach for performing service traffic shaping in the presence of network latencies, before validating it through extensive simulations in Section 5.3. Finally, in Section 5.4, we conclude this chapter.

5.1. The Service Traffic Shaping Problem in Geographically Distributed EDCs

In this section, we identify the challenges for performing service traffic shaping in geographically-distributed two-tier EDCs. We first describe the system architecture which will be the basis of our study. A detailed analysis of the existing issues and their consequences comes thereafter.

5.1.1. System Architecture

A service provider may choose to deploy a geographically-distributed EDC. Besides being able to provide services for different customers in multiple locations, other business motivations may be involved when choosing to implement such setups. For example, around 15% of the operating costs of a datacenter correspond to electricity costs[15]. A service provider may therefore want to lower its operation costs by locating its service hosts in a geographical location where the electricity is cheaper[28].

For our current study, we instantiate the system architecture as a geographically-distributed EDC with two main entities: Preprocessing clusters, noted C_i, formed by SON appliances, and service instances (S_m), as depicted in Fig. 5.1. We assume that there are G different preprocessing clusters accessing concurrently H different

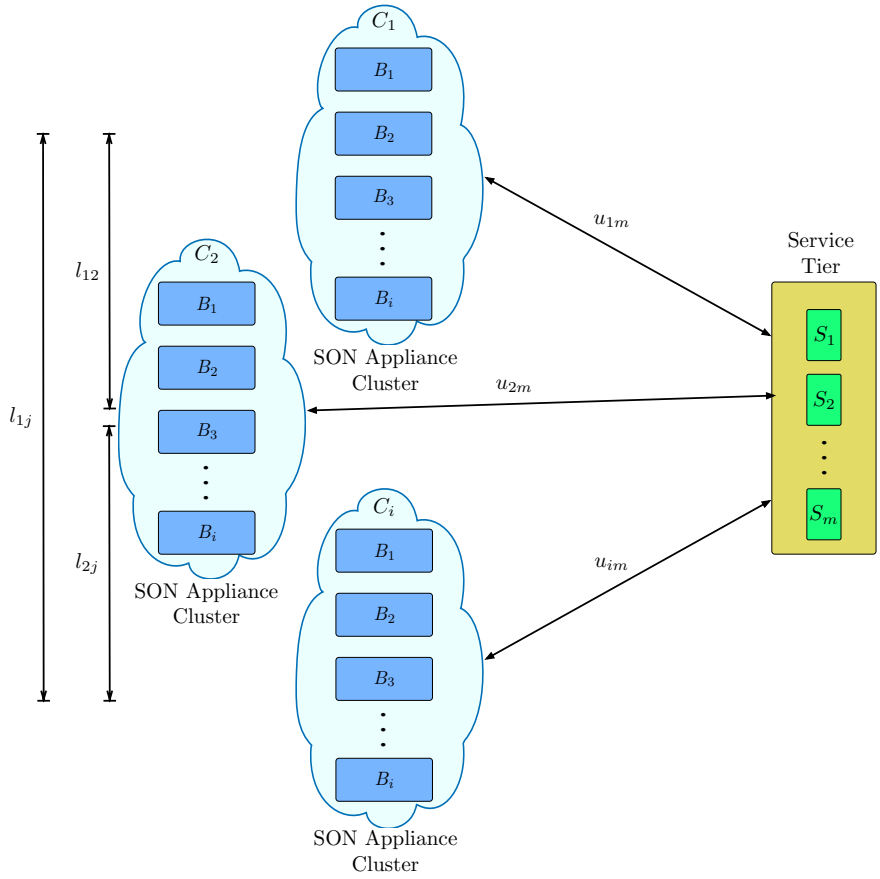


Figure 5.1: System Architecture showing the different elements involved in the service contract enforcement procedure for the *geographically-distributed* case. Bidirectional arrows represent symmetric one-way communication delays.

service instances on the service tier. Each one of the entities (clusters and service instances) is located in a different geographical location. Communication delays therefore exist between clusters themselves, and between clusters and services. One-way communication delays between clusters i and j are noted l_{ij}. Similarly, one-way communication delays between a cluster i and a service m are noted u_{im}. All delays are symmetrical, such that $l_{ij} = l_{ji}$ and $u_{im} = u_{mi}$.

5.1.2. Issues in Geographically Distributed EDCs Inherent to the Presence of Network Latencies

Existing service traffic shaping approaches are designed specifically for single-site EDCs. In this kind of deployments, usually the entities are located either located on the same rack or in the same server room, with high speed links interconnecting them. Therefore, existing shaping solutions consider that the communications delay between all involved entities is negligible.

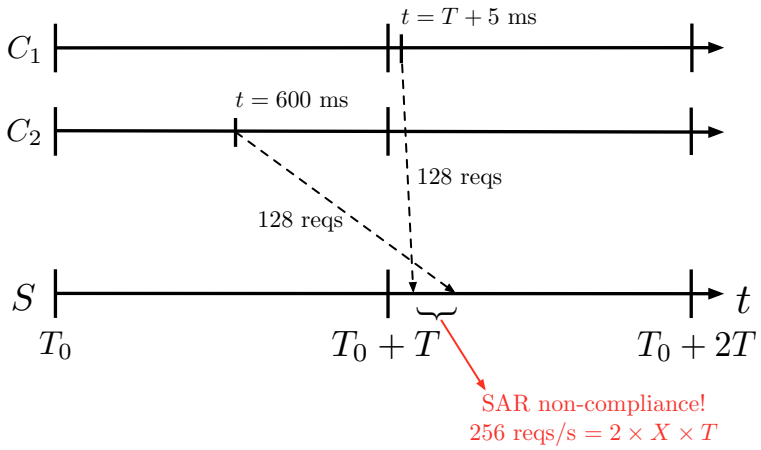


Figure 5.2: Timeline illustrating the *late request arrival* issue. Because of network delays, the service instance receives two times the maximum allowed rate specified by the SAR during one enforcement period, thus leading to SAR non-compliance.

Nevertheless, in geographically-distributed environments, a main issue arises when applying current techniques designed for single-site EDCs. A common drawback found in existing static or dynamic approaches is what we call *late request arrival*. Because of communication delays, the existing service traffic shaping algorithms may lead to SAR non-compliance since requests leaving the appliance at one enforcement period will arrive at the service during the next one.

To better illustrate this issue, consider the following example. Denote by $X \times T = 128$ docs/s the SAR specified by the IT administrator, $T = 1$ s the SAR enforcement period, and $u_{11} = 10$ ms and $u_{21} = 500$ ms the communication delays from clusters C_1 and C_2 respectively towards a service host S_1. Suppose that during subperiod T, cluster C_2 is the only one which receives requests, and it receives 128 requests to process at around $t = 600$ ms. It is allocated all the available credits, and depending on the processing time, it will send 128 requests to the service instance. Next, suppose that during subperiod $2T$, cluster C_1 is the only one which receives requests to process, and it receives 128 requests around $t = T + 5$ ms. After preprocessing them, it will send 128 requests to the service host a few moments later. Because of network latencies, the service host will receive 128 requests around $t = T + 15$ ms, and 128 requests around $t = T + 100$ ms. Therefore, during the same enforcement period, the service host would have received $2 \times X \times T$, which is twice the rate allowed by the SAR. Fig. 5.2 depicts the timeline for the exchanges described above.

5.1.3. Discussion

Because of the network latencies introduced by geographical distribution, current approaches, specifically designed for single-site EDCs, may lead to SAR non-compliance under certain conditions. Therefore, it is clearly necessary to design new algorithms that are able to solve the service traffic shaping problem in the presence of non-negligible network latencies, while still being able to comply with the SAR.

5.2. GEODS: Geographically Distributed Service Traffic Shaping

To address the new challenges imposed by communication delays inherent in geographical distribution, we propose GEODS, a geographically-distributed service traffic shaping algorithm. We mainly undertake three actions. First, in order to tackle the late arrival issue, we propose that each SON appliance cluster accessing the service tier must have a *sending deadline*, which is the latest time at which a cluster may send requests to the service host. Second, in order to dynamically adapt to changes in the input rates to the system, and tackle latency related issues, we propose that the number of subintervals used during each subperiod, must depend on the latencies between the clusters and the service hosts. Finally, in order to reduce issues due to outdated information, the number of information exchanges during each subperiod must be minimized. In this section, we first specify the preliminaries of our approach. The details of GEODS come thereafter.

5.2.1. Assumptions

Without loss of generality, we make the following basic assumptions. First, all clusters in the system know their one-way communication delays with the service hosts, with each other, and they also know these values for the other clusters in the system. Furthermore, for clarity's sake, we suppose that all these delays are symmetrical, meaning that they have the same value in one direction as they have in the other direction. In other words, the value of the delay between a cluster and a service host will be equal to the value of the delay between the said service host and the cluster. Moreover, there is little or no variation in the value of this delay. Many service providers, such as Verizon, guarantee that the jitter between entities will not exceed 1 ms[37]. Finally, for simplicity, we assume that all entities in the system are

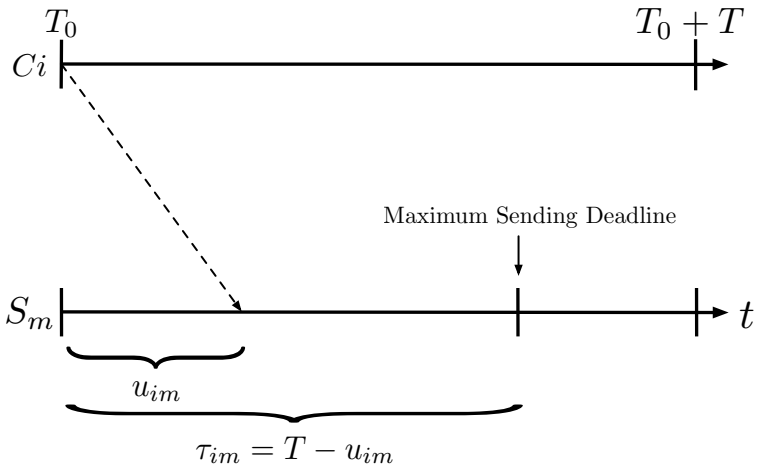


Figure 5.3: Timeline depicting the *maximum sending deadline* τ_{im}.

synchronized, meaning that the enforcement period will start at the same time in all locations.

5.2.2. Choosing the Maximum Sending Deadline

The first step of GEODS is the choice of the maximum sending deadline (see Fig. 5.3, or the time limit at which each cluster will be able to send requests to a service service host. This value is not a "global" value as each cluster will have its own deadline for each service. For cluster C_i, we note this value τ_i. Its calculation is as follows:

$$\tau_{im} = T - u_{im}, \tag{5.1}$$

where T is the enforcement period, and u_{im} is the communication delay between cluster i and service m. At the start of each enforcement period, each cluster will calculate its own value of τ_{im} for each service. After this deadline has been reached, no further requests can be send to the service hosts even if there are credits still available.

5.2.3. Adapting to Changes in Incoming Traffic

In order to dynamically adapt to variations in the rate of incoming traffic entering the system, GEODS relies in the notion of enforcement subperiods introduced earlier in this manuscript (see Chapter 2). However, we added some modifications in order to accommodate the new constraints induced by geographical distribution. At the beginning of each enforcement period, each cluster will calculate T_m^*, which is the

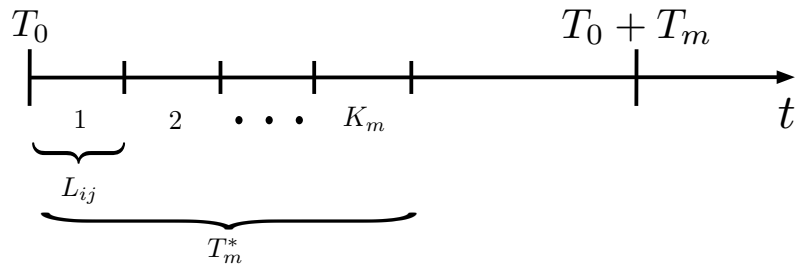


Figure 5.4: Timeline illustrating the different parameters involved in the credit allocation procedure for GEODS.

lowest value of the lowest sending deadlines for service m; and L_{ij}, which is the largest network latency between any pair of clusters in the entire system:

$$T_m^* = \min[\tau_{im}], \tag{5.2}$$

$$L_{ij} = \max[l_{ij}]\cdot \tag{5.3}$$

The idea behind the calculation of these values is to find the maximum number of enforcement subperiods that can be used to readapt to the conditions of the traffic as often as possible. Because of network delays between clusters, in order to perform calculations with up-to-date information, exchanges between clusters can only happen as often as the largest latency between clusters allows, in other words, every L_{ij}. Furthermore, the calculation of T_m^* is necessary since, for the cluster with the lowest sending deadline, after this value has been reached, the said cluster is no longer able to send requests to the service host. Therefore, further information exchanges with this cluster should not be made. Note that, clusters with sending deadlines closer to the value of T would still be able to send requests to the service instances, and would therefore still be able to adapt to the changing conditions of incoming traffic. However, by design choice, all clusters in the system will no longer perform information exchanges after T_m^* has reached. Clusters with sending deadlines later than T_m^*, however, will still be able to send requests to the service hosts if they have credits still available.

A timeline which depicts all the involved parameters is found in Fig. 5.4.

Subsequently, the number of subperiods for a given service, noted K_m, is calculated in the following way:

$$K_m = \left\lfloor \frac{T_m^*}{L_{ij}} \right\rfloor \tag{5.4}$$

Table 5.1: Verizon Business round trip communications delay guarantees. [37]

Location	Europe	North America	Asia
Europe	30 ms	110 ms	250 ms
North America	110 ms	45 ms	230 ms
Asia	250 ms	230 ms	125 ms

Once all of these values have been calculated, all clusters initiate the credit allocation procedure.

5.2.4. Performing credit allocation

GEODS uses the same credit allocation heuristic proposed in DOWSS (cf. Chapter 3), but adapted to handle multiple services as described in Section 4.2.2. At the beginning of each enforcement subperiod, each cluster will broadcast to the others a *single* message containing all the information necessary to perform credit allocation. The message includes the following values:

- r_{im}: the number of requests cluster i sent to the service host m during the previous subperiod.
- Q_{im}: the sum of the queue sizes of all appliances in cluster i for service m.
- N: a random generated number which will be used to solve SAR non-compliance.

With this information, all clusters in the system will be able to *locally* calculate all the values necessary for the credit allocation procedure, such as, cluster weights, and the number of requests remaining during the subperiod. By minimizing information exchanges between clusters, we ensure that all the operations involved in the credit allocation procedure are done using up-to-date information.

5.3. Evaluation

To study the performance of GEODS, we undertook a series of simulations. To this end, we used the OMNeT++ Discrete Event Simulation System [35]. The OMNeT++ library controls the simulated time and the concurrent execution of the code running on each one of the simulated SON Appliances. All appliances run the same code. The algorithms are written in C++ and are event driven. The simulation-based evaluation of our proposal is centered around answering three questions:

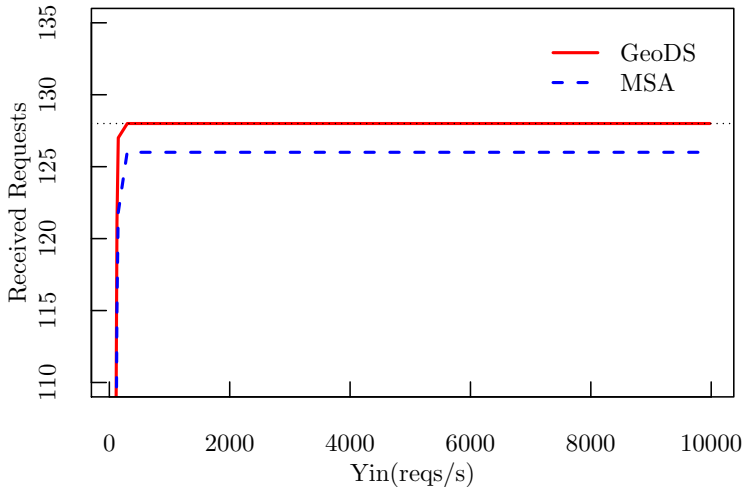


Figure 5.5: Comparison of the performance between a static allocation approach and our approach for geographically-distributed setups for different input rates to the system.

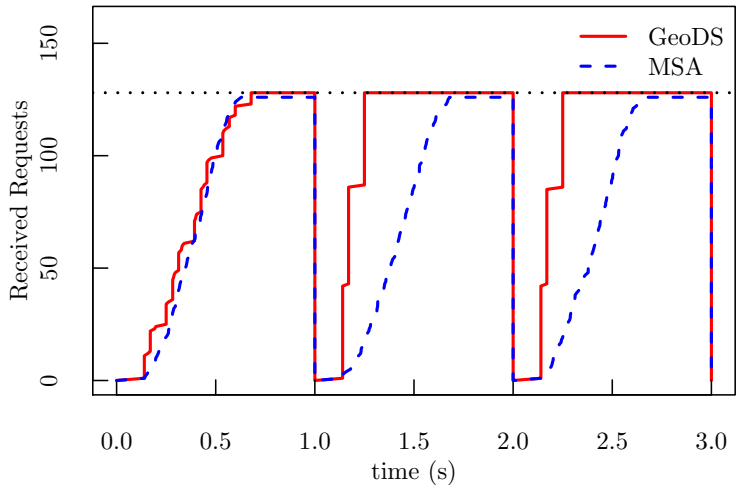
1. Is our approach able to solve the service traffic shaping problem in geographically-distributed environments?

2. Are the techniques used by our approach better than techniques applied in single-site production environments?

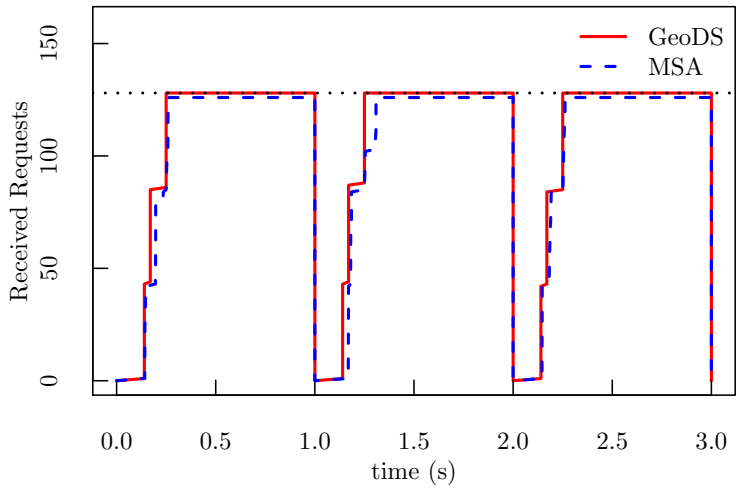
3. Will the our algorithm be able to prevent the *late arrival* issue while continuing to comply with the SAR?

To answer questions (1) and (2), we modeled client service requests as Poisson processes. The *average* input rate to the system, noted as Y_{in}, is chosen as a fixed value unknown to the SON appliances clusters. In order to verify SAR compliance for all input rates this value is varied. Representative results are shown in Fig. 5.5 and 5.6 and are discussed later in this section. Next, we proceed to answer question (3). To do this, we simulated Poisson traffic which we deliberately delayed by a fixed value, so that the first requests arrive at clusters after the first half of the first enforcement period. Representative results are shown in Figs. 5.7 and 5.8.

We assume that the processing rate of each document at each appliance varies and depends directly on request sizes. For simplicity, we chose to perform simulations for a single service. Nevertheless, the algorithm has been designed for handling multiple services. For all the presented simulations, we have set $T = 1$ second, $C = 6$, and $X \times T = 128$ docs/s. We defined the geographical locations for all entities according to the round trip communication delays guaranteed by Verizon's SLA[37] (see Table 5.1). For our simulations we chose the following geographical locations: C_1 is located in Europe C_2 is located in North America, C_3 is located in



(a) $Y_{in} = 300$ reqs/s



(b) $Y_{in} = 10000$ reqs/s

Figure 5.6: Performance of GEODS for different input rates to the system during 3 enforcement subperiods.

Asia. The service host S is located in Europe. All data points shown on the curves represent an average over 50 runs.

In Fig. 5.5, we explore the performance of both GEODS and MSA during an enforcement period. This figure shows the number received at the service host during one enforcement period, as a function of the input rate. The horizontal dotted line shows the value of $X \times T$. For sending rates much lower than SAR, both schemes perform equally, as expected. However, for sending rates over $X \times T$, our algorithm outperforms MSA. Because of the approximations functions used for credit allocation in MSA, even though it closely approaches the value of $X \times T$, it never reaches it. On the other hand, our approach pushes the system to its maximum by pro-

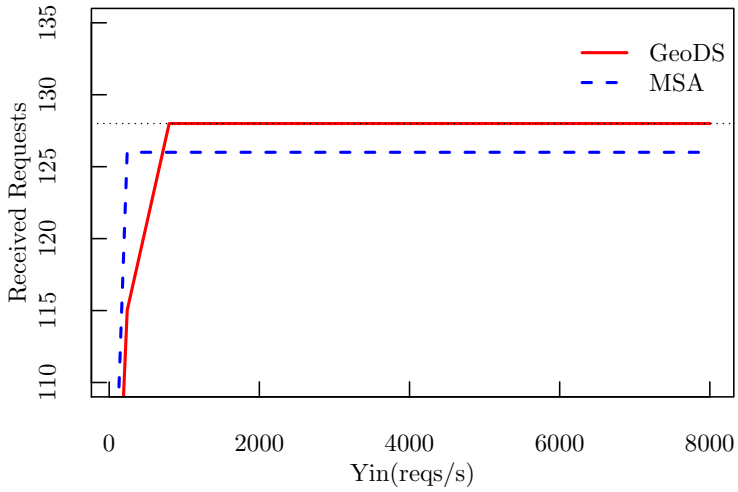
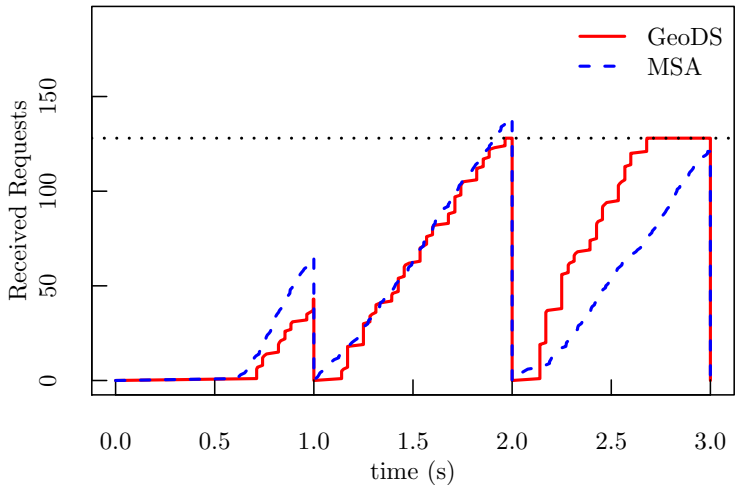


Figure 5.7: Comparison of the performance between a static allocation approach and our approach for geographically-distributed setups for different input rates to the system, when request arrivals start late in the subperiod

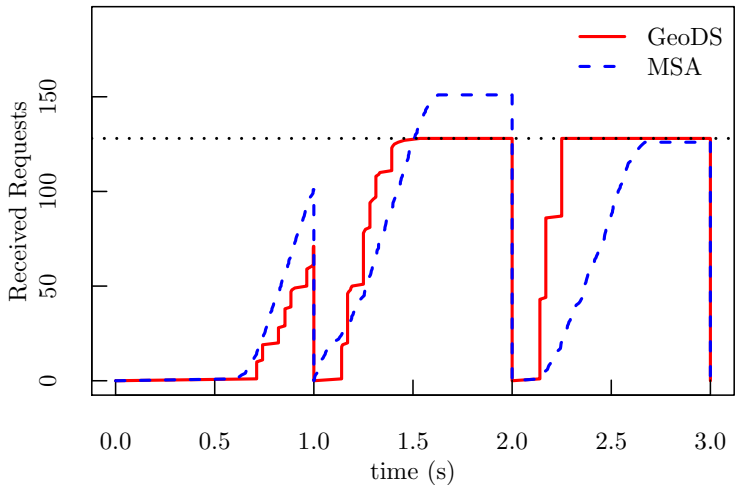
cessing and sending exactly X requests during the observation period to the service host.

In Fig. 5.6, the performance of both GEODS and MSA over three observation periods is depicted. This figure shows the number requests received at the service host during three enforcement periods. In Fig. 5.6(a), Y_{in} is set to 300 requests/s. In this case, contrary to MSA, our algorithm is able to fully utilize the system's available resources sending exactly $X \times T$ requests to the service hosts. In Fig. 5.6(b) Y_{in} is set to 10,000 requests/s. Even with a high input to the system, our algorithm is able to comply with the SAR established by the IT manager. Note that, in both cases, there is an initial warmup period with a duration of $\max[l_{ij}]$ during which no requests are treated. This is further evidenced later in the period as during each information exchange, no requests are treated. Therefore, once credits are allocated, requests leave the clusters as quickly as possible. On the other hand, MSA is also able to comply with the SAR because it is actually able to use all the available credits early in the enforcement period. Therefore, no requests sent during one period arrive during the next one.

In Fig. 5.7, we explore the performance of both our algorithm and MSA when the input traffic starts arriving half way into the enforcement period. The figure shows the number requests received at the service host during one enforcement period, as a function of input rate. The horizontal dotted lines show the values of $X \times T$. Even, if requests start arriving late in the period, our algorithm is able to comply with the SARs. Nevertheless, for input rates close to the service contract, the service host



(a) $Y_{in} = 150$ reqs/s



(b) $Y_{in} = 300$ reqs/s

Figure 5.8: Performance of the algorithm for different input rates to the system during 3 enforcement subperiods when request arrivals start late in the subperiod.

receives less request than the maximum specified by the SAR. Interestingly, MSA is able as well to comply with the SAR, but as before, it is not able to send the maximum allowed by the service contract. Both of these behaviors shown on the curves occur because of the way our simulation results were obtained. Since system starts sending requests late in the first period, for Y_{in} values around $X \times T$, the actual number of requests entering the maybe less than the maximum number of requests specified by the SAR. Moreover, obtained curves are averages obtained from 3 second runs, it appears that MSA is able to fully comply with the SAR. However, this is not the case as it will be shown in the following.

In Fig. 5.8 we explore the performance of both our approach and MSA during 3 enforcement periods, when requests start arriving late in the first period, for both low and average entry rates to the system. In both cases, during the first period, both approaches perform similarly and send less requests to the service host than the specified contract. This is due to the actual number of requests entering the system. However, during the second period, MSA clearly violates the SAR, by sending to the service host, more than $X \times T$ requests. Indeed, since towards the end of the period the algorithm still has available credits, it will send them. These requests will arrive during the next enforcement period, thus leading to SAR non-compliance. In contrast, GEODS is able to strictly comply with the SAR. Because GEODS defines a *sending deadline* request sending so that requests do not overlap into the next period, it avoids the *late arrival* issue, while still sending exactly $X \times T$ requests to the service instance.

5.4. Conclusion

In this chapter, we studied the service traffic shaping problem in *geographically-distributed* EDCs. Current service traffic shaping approaches, which have been designed specifically for single-site setups, face performance issues, namely *late request arrivals* when in the presence of network latencies inherent to geographically-distributed deployments. To tackle the network delay-related issues inherent in geographical distribution we propose GEODS, a geographically-distributed service traffic shaping algorithm, which takes into account in its design the communication delays between all entities in the system. By introducing a *sending deadline*, calculated using latency information, our approach is able to make credit allocation and request sending decisions, while preventing the current issues which lead to SAR non-compliance. Our simulation results show that, GEODS is able to efficiently solve the service traffic shaping problem in geographically-distributed EDC setups, while being able to strictly complying with the maximum sending rate established by the SAR.

Chapter 6

Conclusions and Perspectives

The Internet changed the way business is conducted worldwide. To remain competitive, businesses have been implementing information technology support for business processes over the years. In this context, Service Oriented Architectures (SOA) have emerged as the main solution for the integration of legacy systems with new technologies within large organizations. Modern Enterprise Data Centers (EDCs) implementing SOA concepts and solutions are usually deployed as a two-tiered architecture where, in order to relieve service servers from the computational cost of CPU intensive tasks (e.g., XML parsing) and to perform resource protection, these functions are offloaded on a cluster of SON (Service-Oriented Networking) Appliances. In EDC setups, access to services is governed by Client Service Contracts (CSCs) dictated by Service-Level Agreements (SLAs), which aim at protecting EDC resources, which are usually CPU power and main memory from application servers, disk storage from storage servers, and link bandwidth on the internal EDC network. Currently, SON appliances are only able to protect EDC resources by limiting the access (i.e., controlling the traffic) to services. Since in an EDC environment, service clients access services from *multiple* entry points (e.g., a cluster of SON appliances), the desired effect is "global" shaping. *The challenge is then to enforce contracts by taking local actions at each entry point.* In this thesis, we proposed three contributions in this field.

6.1. Contributions

6.1.1. DoWSS: Multipoint-to-Point Service Traffic Shaping

First, we proposed DoWSS, a doubly-weighted algorithm for service traffic shaping in service-oriented networks. Similar to existing approaches, our solution relies on the notion of *credits* which allow appliances to send requests to the service tier, and the subdivision of the Service Access Requirements (SAR) enforcement period into *subperiods*. However, unlike existing credit-based approaches, our approach guarantees the allocation of at least one credit per measurement subperiod by effectively solving the numerical approximation issues. DoWSS implements a doubly-weighted strategy for credit allocation, using weights based on request sizes. This aims at penalizing the appliance queues that would take the longest to process, by assigning more credits to appliances with smaller queues, thus preventing *starvation*. The performance of DoWSS was evaluated via extensive simulation. Our results showed that DoWSS performs optimally by processing exactly $X \times T$ requests per observation period, which is the maximum possible number of requests allowed by the SAR.

6.1.2. MuST: Multipoint-to-Multipoint Service Traffic Shaping

As next step, we stated that current off-the-shelf appliances present architectural limitations that prevent them from efficiently performing service traffic shaping towards *multiple* service instances. After clearly identifying these limitations, namely the use of a single exit FIFO queue, we introduced MuST, an approach aimed at addressing current architectural limitations in order to efficiently perform *multi-service* traffic shaping. Our approach proposes a novel SON appliance internal architecture based on the use of one exit queue per service present in the service tier. MuST was validated through thorough simulation analyses. The obtained results prove that MuST efficiently solves the multipoint-to-multipoint service traffic shaping problem while pushing the system to its maximum capacity.

6.1.3. GeoDS: Geographically Distributed Service Traffic Shaping

Finally, we argue that current service traffic shaping approaches face performance issues when they are used in geographically distributed setups, since they have been specifically designed for single-site EDCs. To tackle the network delay-related issues inherent in geographical distribution, which we refer to as *late request arrivals*,

we propose GEODS, a geographically distributed service traffic shaping algorithm. Our approach is based on the knowledge of the communications delays between all entities in the system, and introduces the notion of *sending deadline*, which is the maximum time at which a cluster of SON appliances may send requests to a service host. Our simulation results show that, GEODS prevents the *late request arrivals* issue, and is able to strictly comply with the maximum sending rate established by the SAR, thus efficiently solving the service traffic shaping problem in geographically distributed EDCs.

6.1.4. Conclusion

Nowadays, services offered by companies all over the world often take the form of distributed software applications hosted in enterprise data centers. In order to remain cost effective, the availability of shared resources of an EDC must be guaranteed. In two-tier setups, the preprocessing tier protects EDC resources by limiting the number of requests per second sent to a service instance. In this thesis, we proposed three solutions to perform service traffic shaping in this setups to address some of the challenging issues imposed by this environment, such as the presence of multiple entry points, multiple service instances, and geographical distribution. Overall, our approaches are able to efficiently solve the service traffic shaping problem while pushing service utilization to the maximum.

6.2. Perspectives

6.2.1. Lack of Standardized SAR

One of the major issues of implementing Service Traffic Shaping algorithms in Service-Oriented Networks, is the fact that the SAR are neither standardized, nor well-defined. Indeed, SAR found in current production environments are defined in terms of requests per second as opposed to data units per second as it is done in classic networking. Moreover, they do not include parameters such as a maximum burst size. Consequently, SAR in SON are not precisely measurable, making the resource (CPU processing time in this case) difficult to protect. The challenge is therefore, finding a way to establish other types of SAR that are well-defined.

6.2.2. Different Protection Metrics

Resource protection mechanisms available in current off-the-shelf SON appliances consist of limiting the *number of requests per second* sent to a service instance aiming at protecting CPU power. However, in Service-Oriented EDC environments, the resources protected by the shaping function also include, main memory, storage capacity, and link bandwidth. Since SON appliances do not perform deep content inspection of the requests to be serviced, there can be no estimates on how much CPU power or memory a particular request may require. Finding new ways of protecting EDC resources, containing new protection metrics which are fit for the cloud space, remains an open issue. A short discussion on this issue is found in Appendix B.

6.2.3. Appliance Weights vs. User History

By design choice, our main heuristic calculates appliance weights using request sizes leading to queue sizes as its main metric. Nonetheless, in the future, it could be useful to investigate approaches for assigning weights to appliances, such as creating a history of appliance usage based on user history.

6.2.4. Scalability

This is a fundamental issue in the design and operation of an enterprise network. The scalability of the proposed architectures and algorithms depends on a number of factors such as the SAR, the length of the enforcement period, the number of enforcement subperiods, and the number of services present on the service tier. To this end, EDCs will need to implement scalability metrics. In this work we deliberately did not conduct a study on the scalability of our approaches but, instead we focused on the main details of the proposed architectures as well as the core functionalities of our algorithms.

6.2.5. Distributed Shaping

We have considered a decentralized deployment architecture in which there are several SON appliances accessing a single service host. In this scenario, each appliance runs the proposed algorithm while having a **global** knowledge of the state of the rest of the appliances. To this end, the appliances exchange information with each other. Each appliance calculates locally the sending rate. Nevertheless, there is another

deployment scenario where the algorithm is completely distributed, and each appliance will calculate locally the sending rate towards the service host while having a *partial* knowledge of the state of the rest of the appliances. For our contributions we did not consider this scenario, and focused on the decentralized case.

6.2.6. Real-World Testbed and Datasets

The simulation results presented in the present paper show the strengths and usefulness of the approach, as well as some of its limitations and possible drawbacks. The next step is to design and implement a practical version of the proposed approach on a real world testbed which would allow properly measuring the impact of the algorithm in an actual production environment. Additionally, accounting for the lack of a testbed, the obtention of datasets from actual production environments could also give insights on the performance of our solutions in real world scenarios.

Appendices

Annexe A

Résumé de la thèse en français

PENDANT les 20 dernières années, l'ascension des Technologies de l'Information et de la Communication (TIC) a changé la face du monde. À la fin des années 80, une des conclusions clés du Landmark MIT Study[45] signalait que les TIC étaient devenues une ressource vitale pour être compétitif dans le marché global, ce qui a poussé un grand nombre d'organisations à considérer les TIC comme ne composante essentielle de leur stratégie globale. Par conséquent, pour pouvoir rester compétitives, les entreprises partout dans le monde ont progressivement implementé du support informatique pour leurs processus métier. Associé à ceci, l'émergence du World Wide Web, qui a transformé l'Internet, une réseau utilisée principalement par le monde universitaire et la recherche, en une réseau mondiale qui interconnecte des entreprises et des consommateurs partout. Alimentées en partie par ces avancés technologiques, ont contribué à la modification de la plupart des économies mondiales. Celles-ci, ont cessé d'être basées principalement sur l'agriculture et la manufacture, pour devenir des économies de services, engendrant l'apparition de nouvelles façons de fournir des services comme la banque en ligne, des super magasins de vente au détail hautement efficaces, le commerce en ligne, entre autres.

A.1. Une Internet Orientée Service

Les services en ligne proposées par des entreprises partout dans le monde souvent prennent la forme d'applications logicielles distribuées et hébergées dans des serveurs en arrière plan. La tendance actuelle est de localiser des applications métier dans des centres de traitement de données géographiquement distribués[27], qu'ils soient détenus par les entreprises ou externalisés, qui opèrent comme des nuages informatiques privées ou publiques, de façon à ce que les opérations puissent migrer

entre les sites de façon transparente, en gardant les sessions d'utilisateur, la disponibilité des applications, et l'accès aux données. Une étude récemment publiée par Gartner Research[29] montre que la France est l'un des leaders Européens en termes d'adoption de ce genre d'offres d'hébergement, vu que le 71% des organisations inclues dans le sondage avaient déjà utilisé des logiciels en tant que service (Software-as-a-Service ou SaaS en anglais) pour un ou plusieurs applications métier, tandis que le 29% restant envisageaient de le faire dans les 12 prochains mois suivant l'enquête.

Un tel environnement soulève un nombre de problèmes de gestion, performance, résilience, et sécurité. Néanmoins, un des plus grands challenges affrontés aujourd'hui par les administrateurs des parcs informatiques est l'intégration des applications. En raison de l'hétérogénéité des systèmes, la communication des systèmes patrimoniaux entre eux et avec des systèmes plus modernes, à travers de différents fournisseurs, protocoles, et logiciels, se révèle être une tache difficile. En plus, l'évolution rapide des matériels et logiciels disponibles, amplifie le problème de la garantie du bon fonctionnement des systèmes capables de s'adapter aux exigences métier. Dans ce contexte, les Architectures Orientées Service (SOA)[18] sont devenues la solution principale pour l'intégration des applications et technologies dans le monde de l'entreprise et pour la collaboration entre des partenaires industriels.

SOA est une architecture logicielle pour construire des applications qui implémentent des processus métier ou services en utilisant un ensemble de boîtes noires faiblement couplées, lesquels sont orchestrées pour délivrer un niveau de service déterminé. Cette architecture a été conçue pour être la prochaine génération d'intergiciels pour la résolution directe des problèmes d'hétérogénéité et de changement. Les SOA peuvent être implémentées par le biais de différentes technologies comme des bus de services d'entreprise (ESB) et des services web (WS)[38]. Ces derniers, sont des systèmes logiciels conçus pour fournir de l'interoperabilité entre machines à travers d'un ensemble de standards ouverts et basés sur le langage de balisage extensible (XML), tels que le langage de description de services web (WSDL)[40], le protocole d'accès simple aux objets (SOAP)[39], et la description, découverte, et intégration universelles (UDDI)[24] (voir Fig. 1.1).

L'adoption des standards basés sur XML, les WS, et les SOA, a introduit dans le réseau la conscience des applications. Par exemple, le routage est devenu orienté XML avec la capacité de diriger le trafic basé sur du contenu XML en utilisant des fonctions comme le routage XPath[41]. En outre, actuellement il est possible d'effectuer des décharges fonctionnelles comme la transformation XML (XSLT)[42], qui permet de modifier le contenu XML au fur et à mesure de sa traversé du réseau,

ainsi que la médiation de services pour assurer l'interoperabilité des services web dans des environnements hétérogènes. Pourtant, même si l'utilisation des standards basés sur XML permet une intégration simple avec des sources de données extérieures, un des problèmes principales empêchant une adoption plus étendue des services web est la *performance*[46]. En effet, vu que le temps nécessaire pour l'analyse syntaxique d'un document XML peut être de l'ordre de quelques minutes[17], le temps de réponse d'un service web est potentiellement large. Pour mieux satisfaire les objectifs métiers, les fournisseurs de services utilisent des intergiciels qui fournissent des capacités pour créer des réseaux orientés service (SON)[7], notamment de l'analyse syntaxique de XML accéléré, la décharge fonctionnelle, l'intégration de protocoles, et du routage basé sur le contenu.

Avec l'intégration de ces nouvelles technologies, les offres d'hébergement en nuage d'applications dans le domaine public sont en train de devenir compétitives de manière accélérée. Pour les administrateurs système ceci implique le déploiement et la gestion de centres de traitement bien provisionnés, et prêts à gérer des charges de service en éternel changement. En effet, les applications en nuage, qui nécessitent plusieurs CPUs pour tourner, qui sont potentiellement utilisées par des dizaines de milliers d'utilisateurs tous les jours, posent des exigences énormes sur les ressources du centre de traitement. Dans un centre de traitement d'entreprise, ces ressources sont principalement la puissance de calcul, la capacité de mémoire et de stockage, le débit de requêtes et la bande passante du réseau, lesquels doivent être partagés par toutes les applications hébergées dans le centre de traitement. De plus, pour fournir une infrastructure de calcul fiable et qui passe à l'échelle, les centres de traitement sont provisionnés spécifiquement pour le pire des cas, ou les heures pleines. Dans la plupart des installations, environ le 45% des coûts sont reviennent aux serveurs[15]. Pour amortir ces grands coûts d'investissements requis pour le provisionnement des centres de traitement, il faut s'assurer d'un important taux d'utilisation pour ces derniers. Malheureusement, le taux d'utilisation dans les centres de traitement est remarquablement faible (e.g. 10%)[15]. En raison des longues échelles temporelles pour le provisionnement, la taille des investissements, et le manque de certitude sur la demande, le conservatisme menant au surprovisonnement est un mécanisme naturel pour la gestion de risques. Ainsi, dans les centres de traitement, la protection de ressources a pour cible la réduction des interruptions et l'amélioration de la disponibilité globale des ressources.

A.2. Formulation du Problème : La Protection de Ressources dans des Centres de Traitement d'Entreprise

Le provisionnement et la protection des ressources sont des problèmes classiques liés à la qualité de service. Leur but est de fournir aux clients des garanties de service. Pour mieux respecter les garanties de niveau de service, et pour protéger les ressources partagées du centre de traitement, les administrateurs système établissent des contrats de niveau de service (SLA), lesquels définissent des règles pour le partage de ressources, et fournissent des garanties de qualité de service aux clients[4]. En détail, un SLA spécifie les métriques qu'un client peut utiliser pour surveiller et vérifier un contrat de service. Ces métriques sont harmonisées aux objectifs métiers du client avec des indicateurs de performance du fournisseur de services. Les SLA « classiques » normalement contiennent un compromis entre des fournisseurs de service pour l'exécution d'une tache particulière dans une limite de temps clairement définie (e.g. résoudre des problèmes sévères en moins d'une heure) ou pour maintenir une disponibilité minimale du service (e.g. disponibilité du 99.99%).

Dans la littérature, Il existe de nombreuses initiatives de recherche académique et industrielle orientées spécifiquement vers l'étude et l'implémentation de mécanismes de protection de ressources dans les réseaux dits « classiques » . Plusieurs techniques spécifiques au partage de ressources dans les centres de traitement ont été développées, afin de répondre aux exigences spécifiées dans un SLA particulier. Celles ci comprennent, entre autres, des algorithmes d'ordonnancement[12;22], des algorithmes de mise en forme du trafic[13;25;26], et des techniques de répartition de charges[10;23].

Néanmoins, dans des environnements SON, le problème de protection de ressources est fondamentalement diffèrent. Les SOA sont des systèmes typiquement centralisés dans lesquels un noeud exécute et gère des instances d'un ou plusieurs services. Cependant, pour palier aux possibles problèmes liés au passage à l'échelle, le service centralisé peut être répliqué et les requêtes réparties entre les répliques[21]. D'une manière générale, les centres de traitement modernes orientés service sont déployés en suivant une architecture logique à deux niveaux (voir Fig. 2.1), où les clients sur l'Internet doivent communiquer avec le premier niveau (ou niveau de prétraitement), pour pouvoir accéder aux applications dans le deuxième niveau (ou niveau de services).

Dans ce type de déploiement à deux niveaux, certaines fonctions comme l'analyse syntaxique du XML, et la limitation du taux d'accès au services, sont déchargées sur un cluster d'appliances SON localisé au niveau prétraitement, pour libérer le niveau de services des coûts en termes de calcul liés a ces opérations[11]. Les mécanismes de protection de ressources disponibles dans les appliances SON actuelles visent principalement à protéger la puissance de calcul du niveau de services. Cependant, le seul moyen dont les appliances disposent pour effectuer cette protection de ressources, pour n'importe quel type de SLA, est en limitant le nombre de requêtes par second envoyées à l'instance de service. Nous appelons ce problème, le *problème de la mise en forme du trafic de services*.

Un service est normalement accédé depuis un seul point d'entré, et le trafic de la passerelle vers le hôte de services suit un modèle point-à-point. Les solutions du monde « classiques » des réseaux de paquets/ATM, comme celles mentionnées auparavant, peuvent donc être appliquées. Par contre, dans les environnements SON les clients peuvent accéder à *plusieurs* services depuis *plusieurs* points d'entrés. L'existence de points d'entrés multiples peut être dictée par la politique de sécurité (la présence de plusieurs zones de sécurité), la robustesse (tolérance aux failles), ou des exigences de performance (la charge du prétraitement est distribuée dans un cluster d'appliances) ; l'effet désiré est une mise en forme « globale » du trafic. *Le défi est donc faire de respecter les exigences du SLA en prenant des actions locales dans chaque point d'entré*.

Dans les réseaux classiques, les ressources protégés par fonction de mise en forme sont typiquement la bande passante du réseau et la taille des mémoires tampon, dont les unités sont définies et peuvent être mesurées avec précision. Les SLA sont standardisés par les partenaires industriels et très clairement définis. Dans les SON, la ressource à protéger par la fonction de mise en forme est la puissance de calcul. En outre, à présent, *les SLA ne sont pas standardisés, et donc non clairement définis et ne peuvent pas être mesurés avec précision*. Par exemple, dans les centres de traitement déployés sur deux niveaux, les administrateurs système établissent des exigences d'accès aux services (SAR), qui visent à protéger le niveau de services du débordement. Normalement, la définition du SAR prend le format suivant :

SAR : « Limitez le taux d'accès à un fournisseur de services à moins de X requêtes par second avec une période d'observation de T seconds. »

Où une période d'observation est une intervalle de temps durant lequel la somme des requêtes envoyées au niveau de services par toutes les appliances ne peut pas dépasser $C = X \times T$. Dans ce cas particulier, vu que la définition de « requête » est faite en unité de requêtes XML, le temps de traitement n'est pas connu exactement. Dans la Fig. 1.3, nous montrons le temps nécessaire pour le prétraitement des requêtes XML en utilisant du trafic réel capturé dans une ferme de serveurs web. En raison de la haute variabilité des délais requis pour l'analyse syntaxique des requêtes en format XML, et même si les passerelles de la bordure du centre de traitement effectuent éventuellement une répartition de charges, ceci ne garanti pas que le taux auquel les requêtes sont envoyées vers le niveau de services soit également réparti.

Cette thèse vise à résoudre efficacement le problème de la mise en forme de trafic de services entre les appliances SON en entrée et les instances de service, tout en considérant les contraintes introduites par les définitions existantes des SAR, les deux étant utilisés dans les systèmes de production actuels.

A.3. Contributions de Cette Thèse

Étant donné les coûts et problèmes inhérents à l'implémentation et provisionnement des centres de traitement, il est impératif de trouver des moyens pour optimiser l'utilisation globale du système. Nous affirmons que pour pouvoir réaliser de la protection de ressources dans des centres de traitement de manière efficace,afin que les déploiements actuels satisfassent les objectifs métiers tout en restant rentables, de nouvelles architectures, de nouveaux contrats de service, et de nouveaux protocoles doivent être soigneusement conçus. Plus en détails ci-dessous, sont présentées les contributions de cette thèse.

A.3.1. Protection de Ressources dans des Centres de Données sur un Seul Site

Dans la plupart des centres de données, la latence n'est pas un problème car l'infrastructure est stockée généralement dans le même rack, ou alors, est aménagée dans la même localisation géographique. Dans ce type de déploiements, les administrateurs système définissent des exigences d'accès aux services, notamment le non dépassement d'un taux maximal d'accès à un service, et le limiter à moins d'un nombre de requêtes de service durant une période d'observation. Dans cette thèse,

nous proposons et validons DoWSS, un algorithme doublement pondéré pour la mise en forme de trafic de services entre plusieurs points d'accès et un seul service. Nous montrons, par le biais de simulations, que DoWSS possède plusieurs avantages : Il élimine les problèmes liés aux approximations de calcul, évite la famine, et maitrise les problèmes liés à la consommation rapide de crédits, qui existent dans les approches actuels basés sur des crédits.

Par la suite, nous affirmons que les appliances SON actuelles présentent des limitations architecturales qui les empêchent d'être utilisées pour effectuer de la mise en forme de trafic d'une manière plus efficace vers *plusieurs* hôtes de service. Dans cette thèse, nous proposons MuST, une architecture interne des appliances SON adéquate à la mise en forme de trafic *multi-service*. Nous montrons par des simulations que notre approche résout le problème de la mise en forme de trafic multipoint-à-multipoint tout en poussant le système à être utilisé à sa capacité maximale.

A.3.2. Protection de Ressources dans des Centres de Données Géographiquement Distribués

Finalement, actuellement les applications sont souvent déployées dans des centres de données géographiquement distribués. Ce type de systèmes introduisent des nouvelles contraintes, en plus de celles déjà présentes dans les systèmes aménagés sur un même site, comme des délais de communication non négligeables. Pour palier à ces problèmes nous proposons et validons GeoDS, un algorithme pour la mise en forme de trafic de services dans des environnement géographiquement distribués. Nous montrons, par le biais de simulations, que notre approche empêche les problèmes inhérents à la présence de latences réseau, et résout efficacement le problème de la mise en forme de trafic de services.

Appendix B

Two Architectures for Resource Protection

CURRENTLY, SON appliances are able to protect EDC resources by limiting the *number of requests per second* a server receives. In practice, the traditional architecture used in EDCs fails when clients define other metrics such as CPU and memory usage. In this annex, we present architectural considerations for performing efficient resource protection under such challenging environments. We argue that, to properly perform resource protection under a large variety of metrics, the system architecture must include new entities. We also estimate, depending on the type of resource to protect, the maximum utilization the said resource may achieve.

B.1. A System Architecture According to the Resource to Protect

So far, because of the system architecture used in this kind of EDC deployments, the preprocessing tier has little or no knowledge of the workload of the service tier. Nevertheless, it is possible to properly protect the resources of an EDC by shaping the traffic that goes from the preprocessing tier towards the service tier. In this section, we classify EDC resources in two classes based on their characteristics and propose two types of two-tier architectures that are able to properly protect them.

We classify EDC resources into two different categories, based on the metrics used to measure their utilization, and therefore how they can be controlled: (i) *Network* resources, for resources with metrics such as request throughput and used link bandwidth; and (ii) *Service* resources, whose metrics include the instantaneous ser-

vice tier utilization in terms of CPU, memory, storage, and the overall state of the internal EDC network. As discussed earlier in this thesis, in order to relieve the service tier from CPU intensive tasks such as XML processing and resource protection, in this architecture those functions are offloaded on a cluster of SON Appliances composing the preprocessing tier. Nevertheless, current SON appliances are only able to protect EDC resources by limiting the number of service requests that are sent to the service tier.

In order to properly protect the other EDC resources by performing service traffic shaping, we propose two different system architectures. The composing elements of each architecture remain the same in both cases; the amount and type of data exchanged, however, differ depending on the type of resource to protect. We introduce two new logical entities into the architecture. The *statistics collector* is the entity in charge of gathering statistical utilization data from the system. The *service traffic shaper* is responsible for using the information collected by the statistics collector in order to properly enforce the service access requirements by performing service traffic shaping, and thus protecting the resource(s) specified by the SAR(s). This entity should implement a service traffic shaping algorithm suited for resource protection under changing network and/or server workload conditions.

The type of resource to protect will determine the type of architecture to use. Network resources can be protected by implementing the system architecture as shown in Fig. B.1 as no information required for performing service traffic shaping is retrieved outside the preprocessing tier. Indeed, in this kind of architecture, the *statistics collector* can keep track of the number of requests sent and overall network utilization by periodically taking measurements at the exit point of the cluster of SON appliances. Therefore, there is no need for the statistics collector to be connected to the service tier.

On the other hand, resource protection for service resources need a closed loop architecture. The interconnection of the above-mentioned entities in this kind of architecture is depicted in Fig. B.2. As some of the data needed for protecting these type of resources (namely instantaneous CPU, memory, storage utilization statistics, and the overall state of the internal EDC network) resides inside the service tier or the internal EDC network respectively, the statistics collector should be connected to both of these entities in order to retrieve it. Such information is exchanged through the internal EDC network, so it is subjected to network delays. Note that both the statistics collector and the service traffic shaper are not hardware entities,

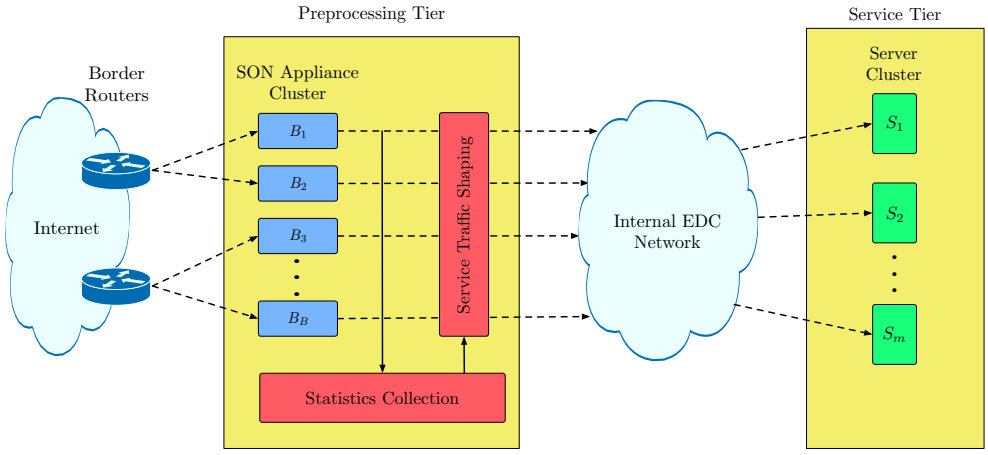


Figure B.1: *Open loop* system architecture for Service Traffic Shaping.

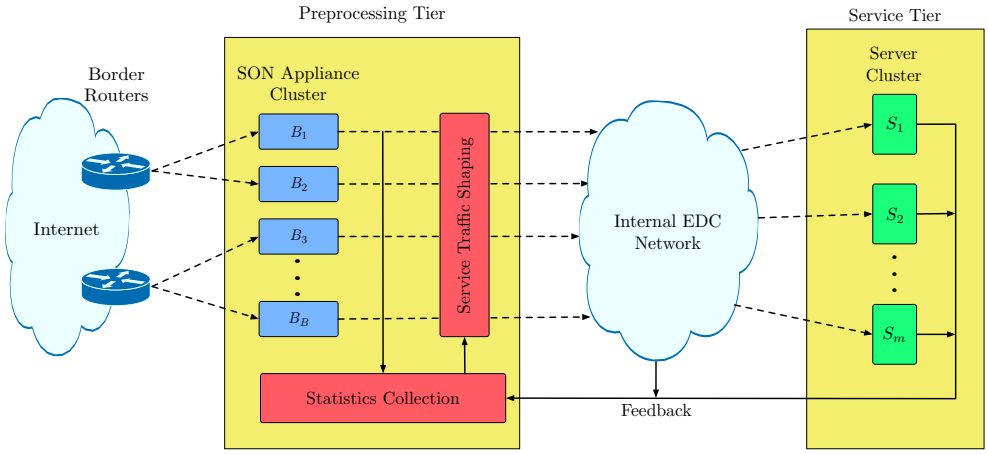


Figure B.2: *Closed loop* system architecture for Service Traffic Shaping.

but rather logical entities that may exist in one or more appliances as a stand-alone or distributed applications.

B.2. The Impact of the Architecture on the Performance of the System

Naturally, resource protection on EDC setups does have an impact on the overall performance of the system. Given the implementation costs and issues inherent to the provision of enterprise data centers, it is imperative to design efficient algorithms that minimize the overall impact on the system while optimizing overall EDC resource utilization. Because of variations on traffic entering the EDC network and

changes in the overall workload of the service tier, dynamic algorithms that adapt to these changes must be conceived.

These algorithms must be capable of using periodically collected resource utilization statistics to better adapt to the system's changing conditions. In other words, it is fundamental that implemented solutions work at their maximum capacity when necessary. However, the reactivity of such algorithms must not compromise the performance of the system. Indeed, since the communication between entities uses the internal EDC network, there is a tradeoff between algorithm reactivity and the amount of overhead it introduces into the EDC internal network. As overall EDC network bandwidth is one of the actual resources to protect, too much information exchange overhead may seriously compromise the performance of the EDC internal network. This is especially true for large EDC deployments containing several thousands of servers.

Algorithm reactivity is also affected by the communication delays inherent in the internal EDC network. In the case of resource protection for service resources, the information that can be obtained is, in general, instantaneous utilization information. Because of network related delays, by the time this information reaches the statistics collector, it may already be outdated. This compromises the performance of algorithms aiming to protect these resources. For geographically distributed EDCs, this represents an interesting challenge, as communication delays may be greater.

The kind of resource to protect also has an impact on the performance of service traffic shaping algorithms. When performing service traffic shaping for protecting network resources, because the utilization of the said resource is easily measurable and quantifiable by using the current metrics available on SON appliances, the system is able to work in a strict mode by taking actions directly on the preprocessing tier that would have almost instantaneous repercussions on the overall performance of the system. On the other hand, existing algorithms designed specifically for this kind of architecture demonstrate that the rate established by the SAR could eventually be respected up to 100% and thus the system could potentially be utilized to its maximum capacity[5;19], as shown in Chapters 3 and 4. On the other hand, when protecting service utilization resources via service traffic shaping, algorithms may not be able to strictly enforce the rate established by the SAR. Indeed, since it is difficult to make a direct equivalence between the number of requests sent to the server and the amount of CPU power, memory, or storage a request will require, resource protection can only be performed inside a set of pre-established soft upper and lower bounds close to the maximum system capacity.

An additional difficulty is that usually multiple appliances should shape traffic towards multiple service hosts. In order to be able to respect the SAR, the preprocessing tier must shape the *aggregate* of the traffic from all appliances in the cluster in a semi-distributed fashion by taking *local actions at each* appliance. Moreover, each different service host usually defines its own service access requirements. On top of that, perhaps not all of the appliances in the service cluster will be able to process requests for all types of service. These particular characteristics found in two-tier EDC setups, raise interesting algorithm design challenges for efficiently performing resource protection by traffic throttling.

Appendix C

List of publications

C.1. Journals

Yesid Jarma, Keerthana Boloor, Marcelo Dias de Amorim, Yannis Viniotis, and Robert D. Callaway. *Dynamic Service Contract Enforcement in Service-Oriented Networks*, IEEE Transactions on Services Computing, preprint, 4 August 2011.

Yesid Jarma, Golnaz Karbaschi, Marcelo Dias de Amorim, Farid Benbadis, and Guillaume Chelius. *Volume-Aware Positioning in the Context of a Marine Port Terminal*, Computer Communications, vol. 34, n. 8, pp. 962 - 972 , June 2011.

C.2. Conferences

Yesid Jarma, Marcelo Dias de Amorim, and Yannis Viniotis. *Towards Multi-Service Traffic Shaping in Two-Tier Enterprise Data Centers*, IEEE CloudCom, November 2011.

Yesid Jarma, Golnaz Karbaschi, Marcelo Dias de Amorim, Farid Benbadis, and Guillaume Chelius. *VAPS: Positioning with spatial constraints*, IEEE WoWMoM, June 2009.

C.3. Posters

Yesid Jarma, Marcelo Dias de Amorim, and Yannis Viniotis, *Multipoint-to-Multipoint Service Traffic Shaping in Three-Tiered Enterprise Data Centers*, Ecole d'été ResCom, June 2011.

C.4. Under review

Yesid Jarma, Marcelo Dias de Amorim, and Yannis Viniotis. MUST*: From Single-service to Multi-service Traffic Shaping in Enterprise Data Centers*, Future Generation Computer Systems, October 2011.

Yesid Jarma, Marcelo Dias de Amorim, and Yannis Viniotis. GEODS*: Shaping Service Traffic in Geographically Distributed Service-Oriented Networks*, IEEE Communications Letters, January 2012.

Bibliography

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM Sigcomm*, Seattle, WA, USA, 2008.

[2] H. Bannazadeh and A. Leon-Garcia. A Distributed Ethernet Traffic Shaping system. In *IEEE Workshop on Local and Metropolitan Area Networks*, pages 1–7, Long Branch, NJ, USA, may 2010.

[3] N Bhatti and R Friedrich. Web server support for tiered services. *IEEE Network*, 13(5):64 – 71, September 1999.

[4] P. Bhoj, S. Singhal, and S. Chutani. SLA management in federated environments. *Computer Networks*, 35(1):5 – 24, 2001. Selected Topics in Network and Systems Management.

[5] Keerthana Boloor, Bob Callaway, Marcelo Dias de Amorim, Adolfo Rodriguez, and Yannis Viniotis. Meeting Service Traffic Requirements in SOA. In *IEEE Workshop on Enabling the Future Service-Oriented Internet*, pages 1 – 5, New Orleans, LA, USA, November 2008.

[6] Keerthana Boloor, Marcelo Dias de Amorim, Bob Callaway, Adolfo Rodriguez, and Yannis Viniotis. Evaluation of multi-point to single-point service traffic shaping in an enterprise network. In *IEEE Globecom*, pages 1 – 6, Honolulu, HI, USA, December 2009.

[7] Robert D. Callaway, Adolfo Rodriguez, Michael Devetsikiotis, and Gennaro Cuomo. Challenges in service-oriented networking. In *IEEE Globecom*, pages 1 – 5, San Francisco, CA, USA, November 2006.

[8] X Chen, P Mohapatra, and H Chen. An admission control scheme for predictable server response time for web accesses. In *International World Wide Web Conference*, Hong Kong, Hong Kong, January 2001.

[9] L Cherkasova and P Phaal. Session-based admission control: a mechanism for improving performance of commercial Web sites. In *IEEE International Workshop on Quality of Service*, pages 226 – 235, London, England, Jan 1999.

[10] T.C.K. Chou and J.A. Abraham. Load balancing in distributed systems. *IEEE Transactions on Software Engineering*, SE-8(4):401 – 412, july 1982.

[11] Gennaro Cuomo. IBM SOA "on the edge". In *ACM Sigmod*, pages 840–843, Baltimore, MD, USA, June 2005.

[12] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Computer Communications Review*, 19:1–12, August 1989.

[13] A. Elwalid and D. Mitra. Traffic shaping at a network node: theory, optimum design, admission control. In *IEEE Infocom*, volume 2, pages 444–454, Kobe, Japan, March 1997.

[14] D Garcia, J Garcia, J Entrialgo, M Garcia, P Valledor, R Garcia, and A Campos. A QoS Control Mechanism to Provide Service Differentiation and Overload Protection to Internet Scalable Servers. *IEEE Transactions on Services Computing*, 2(1):3 – 16, January 2009.

[15] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *Computer Communications Review*, 39:68–73, December 2008.

[16] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. In *ACM Sigcomm*, pages 51–62, New York, NY, USA, 2009.

[17] Michael Head, Madhusudhan Govindaraju, Robert Engelen, and Wei Zhang. Benchmarking XML processors for applications in grid web services. In *ACM/IEEE Conference on Supercomputing*, pages 30 – 30, Tampa, FL, USA, November 2006.

[18] M.N. Huhns and M.P. Singh. Service-oriented computing: key concepts and principles. *IEEE Internet Computing*, 9(1):75 – 81, jan-feb 2005.

[19] Yesid Jarma, Keerthana Boloor, Marcelo Dias de Amorim, Yannis Viniotis, and Robert Callaway. Dynamic Service Contract Enforcement in Service-Oriented Networks. *IEEE Transactions on Services Computing*, preprint, 4 august 2011. doi:10.1109/TSC.2011.45.

[20] K Li and S Jamin. A measurement-based admission-controlled Web server. In *IEEE Infocom*, volume 2, Tel-Aviv, Israel, March 2000.

[21] Vinod Muthusamy and Hans-Arno Jacobsen. SLA-Driven Distributed Application Development. In *MW4SOC*, pages 31–36, Leuven, Belgium, December 2008.

[22] John Nagle. On packet switches with infinite storage. *IEEE Transactions on Communications*, 35(4):435 – 438, April 1987.

[23] L.M. Ni, Chong-Wei Xu, and T.B. Gendreau. A distributed drafting algorithm for load balancing. *IEEE Transactions on Software Engineering*, SE-11(10):1153 – 1161, oct. 1985.

[24] OASIS. UDDI Version 3.0.2, 2004. URL `http://uddi.org/pubs/uddi_v3.htm`.

[25] A Parekh and R Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344 – 357, June 1993.

[26] A Parekh and R Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137 – 150, April 1994.

[27] David A. Patterson. Technical perspective: the data center is the computer. *comacm*, 51:105–105, January 2008.

[28] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. In *ACM Sigcomm*, pages 123–134, New York, NY, USA, 2009.

[29] Gartner Research. User Survey Analysis: Usage Plans for SaaS Application Software, France, Germany and the U.K., 2009. Report, February 2009.

[30] Bianca Schroeder and Mor Harchol-Balter. Web servers under overload: How scheduling can help. *ACM Transactions on Internet Technology*, 6(1):20–52, February 2006.

[31] Amazon Web Services. Amazon EC2 Service Level Agreement, 2011. URL `http://aws.amazon.com/ec2-sla/`.

[32] Cisco Systems. *Cisco Data Center Infrastructure 2.5 Design Guide*. Cisco Systems, San Jose, CA, USA, 2007.

[33] Andrew Tanenbaum. *Modern Operating Systems*. Prentice Hall, 2001.

[34] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems*, 3:1:1–1:39, March 2008.

[35] Andras Varga. The OMNeT++ Discrete Event Simulation System. In *European Simulation Multiconference*, pages 319–324, Prague, Czech Republic, June 2001.

[36] Verizon Business. U.S. Data Center Colocation SLA, 2011. URL `http://www.verizonbusiness.com/terms/us/products/data_centers/premium/`.

[37] Verizon Business. Global Latency and Packet Delivery SLA, 2011. URL `http://www.verizonbusiness.com/terms/global_latency_sla.xml`.

[38] W3C. Web Services Architecture, February 2004. URL `http://www.w3.org/TR/ws-arch/`.

[39] W3C. SOAP Version 1.2 Part 0: Primer (Second Edition), 2007. URL `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`.

[40] W3C. Web Services Description Language (WSDL) 1.1, 2001. URL `http://www.w3.org/TR/wsdl`.

[41] W3C. XML Path Language (XPath) Version 1.0, 1999. URL `http://www.w3.org/TR/xpath`.

[42] W3C. XSL Transformations (XSLT) Version 1.0, 1999. URL `http://www.w3.org/TR/xslt`.

[43] Xiaoying Wang, Zhihui Du, Yinong Chen, Sanli Li, Dongjun Lan, Gang Wang, and Ying Chen. An autonomic provisioning framework for outsourcing data center based on virtual appliances. *Cluster Computing*, 11:229–245, September 2008.

[44] J. Wei, X. Zhou, and C.-Z. Xu. Robust processing rate allocation for proportional slowdown differentiation on internet servers. *IEEE Transactions on Computers*, 54(8):964–977, August 2005.

[45] Ernst & Young. *The landmark MIT study: management in the 1990s*. Ernst & Young, 1989.

[46] S Zilora and S Ketha. Think inside the box! optimizing web services performance today. *IEEE Communications Magazine*, 46(3):112 – 117, March 2008.

[47] M Zukerman, T Neame, and R Addie. Internet traffic modeling and future technology implications. In *IEEE Infocom*, pages 587–596, San Francisco, CA, USA, January 2003.

List of Figures

List of Tables