A Hierarchical Context Dissemination Framework for Managing Federated Clouds

Jeroen Famaey, Steven Latré, John Strassner, and Filip De Turck

Abstract: **The growing popularity of the Internet has caused the size and complexity of communications and computing systems to greatly increase in recent years. To alleviate this increased management complexity, novel autonomic management architectures have emerged, in which many automated components manage the network's resources in a distributed fashion. However, in order to achieve effective collaboration between these management components, they need to be able to efficiently exchange information in a timely fashion. In this article, we propose a context dissemination framework that addresses this problem. To achieve scalability, the management components are structured in a hierarchy. The framework facilitates the aggregation and translation of information as it is propagated through the hierarchy. Additionally, by way of semantics, context is filtered based on meaning and is disseminated intelligently according to dynamically changing context requirements. This significantly reduces the exchange of superfluous context and thus further increases scalability. The large size of modern federated cloud computing infrastructures, makes the presented context dissemination framework ideally suited to improve their management efficiency and scalability. The specific context requirements for the management of a cloud data center are identified, and our context dissemination approach is applied to it. Additionally, an extensive evaluation of the framework in a large-scale cloud data center scenario was performed in order to characterize the benefits of our approach, in terms of scalability and reasoning time.**

Index Terms: **Autonomic communications, cloud computing, semantic context dissemination.**

I. INTRODUCTION

The proliferation of value-added services offered across the Internet has resulted in a significant growth in size and complexity of modern computing and communications infrastructures. It has been argued that current, monolithic management systems will not be able to keep up with this ever-increasing complexity [1]. To alleviate these concerns, researchers have proposed novel network management architectures based on the cooperation between many automated management components, often called autonomic elements (AEs) [2], [3]. Every AE autonomously manages a small subset of the network's resources, while being guided by high-level, human-specified business goals. Obviously, AEs will need to communicate and collaborate in order to achieve their goals. They need to exchange context, which is used to model the current state of their environment, in order to detect sub-optimal behaviour and faults in the underlying network and computing resources. Additionally, as not all problems can be solved locally, it might be necessary for them to initiate management functionality offered by other AEs or negotiate contracts to set up federations across management domains.

Although the increasing management complexity is an issue in many different computing and communication areas, it is especially prevalent in the management of cloud computing data centers. To keep up with the growing demand for cloud computing resources, cloud providers have greatly increased the capacity of their networks and data centers. Additionally, there has been a shift from monolithic single-domain clouds towards large-scale federated cloud environments [4]. This has led to the hypothesis of an Internet-wide inter-cloud, a global cloud of clouds [5]. It is expected these trends will persist in the future, leading to an ever-increasing size, heterogeneity, and complexity of cloud computing infrastructures. Consequently, they have stringent requirements concerning management scalability and efficiency.

This article proposes a framework that facilitates the collaboration and communication between AEs in a large-scale distributed management architecture. More specifically, the presented framework is responsible for the dissemination of huge amounts of context. Context is defined as "the collection of measured and inferred knowledge that describes the state and environment in which an entity exists or has existed" [6]. Additionally, it is detailed how this context dissemination framework fits within a typical autonomic management architecture. The interactions with other important components, such as policies, the knowledge base, federation contracts, and management services are thoroughly described.

Within a management domain, AEs are structured in a hierarchy, which we have previously shown to scale better than flat distributed or centralized approaches [7]. The location of AEs within the hierarchy reflects their management responsibilities. High-level AEs have a wide, yet aggregated, view on the managed environment, and similarly perform high-level management functions. Low-level configuration tasks are delegated to the underlying AEs, which have a more detailed view on a subset of the environment. By restricting the view and/or granularity of the context of AEs depending on the location within the hierarchy, scalability is ensured throughout. The context requirements of an AE obviously depend on its location within the hierarchy. Additionally, these requirements may change dynamically, as the state of the managed environment changes. Existing management approaches often disseminate all context that might be needed by the management algorithms at some point

Manuscript received May 10, 2011.

J. Famaey is funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT-Vlaanderen) under grant no. 73185; S. Latré is funded by the Fund for Scientific Research Flanders (FWO-Vlaanderen).

J. Famaey, S. Latré, and F. De Turck are with the Department of Information Technology, Ghent University-IBBT, email: {jeroen.famaey, steven.latre, filip.deturck}@intec.ugent.be.

J. Strassner is with the Software R&D Laboratory of the US Division of Huawei, email: strazzie@ieee.org.

1229-2370/11/$10.00

in time continuously, which obviously leads to huge amounts of overhead and reduced scalability. To solve these issues, we propose a dynamic context dissemination algorithm that delivers the right context, to the right place, at the right time. The algorithm automatically generates filter rules, which describe what context an AE is interested in. Both these filter rules and the actual context are augmented with semantics. This, in turn, allows AEs to define what context they are interested in, based on the underlying meaning of the information rather than static string patterns. Additionally, it supports the interpretation of received context and thus allows the AE to more accurately perform its automated filter rule generation process.

The management of modern cloud computing infrastructures requires huge amounts of context to be disseminated and processed. Therefore, this management area would greatly benefit from scalable and efficient context dissemination and filtering. Additionally, clouds have specific requirements concerning the filtering and aggregation of context throughout the management hierarchy. More specifically, cloud resource allocation algorithms require specific types of context based on the state of the underlying server resources. To encompass these cloud-specific aspects, and validate our framework for a large-scale management scenario, the presented framework is applied to a cloud management scenario. A prototype of the presented framework was evaluated in detail using a context dissemination scenario in a cloud data center of up to $100,000$ servers.

In summary, this article offers the following novel contributions, which, to our knowledge, are not present in existing context dissemination frameworks. First, a hierarchical context dissemination framework is presented. Context is aggregated and transformed as it is propagated through the hierarchy, offering different views on the managed resources. Second, it not only filters context semantically, based on meaning, but is also capable of dynamically adapting these filter rules to satisfy changing context requirements. Third, the framework is applied to the management of cloud computing data centers. The specific context requirements are identified and it is shown how the context dissemination framework is able to meet them. Finally, by way of an analytical model and an implemented prototype, we evaluate the scalability and applicability of our framework in a large-scale cloud management scenario.

The remainder of this article is structured as follows. Section II lists related work on federated management and the use of semantics in network management. The hierarchical context dissemination between management components in clarified in Section III. Subsequently, Section IV gives a detailed overview of the architectural building blocks that make up the AE and explains how context dissemination fits into it. The introduced architectural concepts are applied to management of cloud environments in Section V. Section VI provides a thorough evaluation and validation of the presented framework, based on an analytical formulation and an implemented prototype. Finally, Section VII concludes the paper.

II. RELATED WORK

This section elaborates on state of the art research related to the work presented in this article. First, recent research on federated network and cloud management is discussed in more detail. Second, the semantic components presented in this article are compared to existing semantic reasoning efforts in network management.

A. Federated Network Management

In recent years, it has been argued that future networks should support loosely coupled collaborations across management domains, or federations. As such, there has been an increasing interest by the research community in federated network management in general [1], [8]–[10] and federated cloud computing in particular [4], [5], [11], [12].

Jennings *et al.* [1] identified several important challenges that need to be tackled in order to achieve a federated Future Internet, capable of supporting loosely coupled end-to-end value networks. Their work addresses six challenges in total, across three levels of abstraction: Federated service management, service monitoring and configuration, and network infrastructure. An important identified requirement is the facilitation of understanding the meaning of measured and inferred data, and supplying the results of these data and their associated conclusions to interested collaborating management entities. Our work builds on that requirement, and provides additional and more concrete semantic reasoning components to accomplish this. Chai *et al.* [8] proposed an orchestration plane that coordinates the interactions within network federations. Their work focusses on a negotiation framework for setting up collaborations between management domains. The proposed work, for now, only supports resource reservations, and is not concerned with the semantics that are needed to achieve understanding between the negotiating parties. Nevertheless, their work can be considered an interesting first step towards a generic contract negotiation framework for setting up dynamic network federations. Feeney *et al.* [10] further study the sharing of capabilities across network domains. In contrast, they do propose a semantic approach, which allows capabilities to be represented in resource description framework (RDF) syntax. They present the layered federation model (LFM), which, among other things, provides a semantic mapping framework. This framework maps diverging semantic models, defined in different management domains, onto one another, which eases the understanding of capabilities across domains.

Recently, the idea of the intercloud, a global federation of clouds, was launched [5]. Its ultimate goal is to allow scaling of applications across multiple cloud providers. This is necessary to achieve the (seemingly) infinite scaling of resource provisioning, as promised by the cloud computing paradigm [13]. Rochwerger *et al.* [11] proposed a modular and extensible cloud architecture for the federation of clouds, called RESERVOIR. It allows providers of cloud infrastructure to dynamically partner with each other to create a seemingly infinite pool of IT resources, while fully preserving the autonomy of technological and business management decisions. More recently, Buyya *et al.* [12] furthered this research towards a federated cloud computing architecture that tackles several pertinent challenges. It is capable of predicting demands and behaviour of the hosted services, and has an economic-model-driven optimization process. Finally, Celesti *et al.* [4] proposed a three stage model for the

evolution of clouds, from the current monolithic cloud providers to horizontal federations, where cloud providers will federate themselves to gain economics of scale and an enlargement of their capabilities. They argue that currently clouds are moving towards the second stage, or the vertical supply chain, where cloud providers leverage cloud services from other providers. We feel that, in order to realize the vision of the intercloud, more semantically rich notions of context, capabilities, constraints, and requirements are necessary. Otherwise, providers cannot efficiently work together, because they do not have the ability to effectively and correctly interpret information and collaborate. Hence, this article proposes semantic reasoning components that enrich the autonomic control loops that govern the providers participating in the intercloud. This provides the appropriate context to enable each provider to make informed decisions when resource and/or service requests are received.

In this article, we present approaches that facilitate both the design and maintenance of a management framework for federated clouds. The design principles used in this framework are based on best practice libraries such as the information technology infrastructure library[1] (ITIL) and enhanced telecom operations map[2] (eTOM). These libraries define high-level best practices that describe processes and process workflows. Specifically for this paper, ITIL aspects such as service operation were taken into account. Similarly, the proposed framework is also based on several eTOM aspects such as resource management and operations and partner relationship management.

B. Semantics in Network Management

Semantics have been widely employed to solve several network management issues, including context dissemination [14] and service matchmaking [15], [16]. The context dissemination process pushes messages, containing context data and associated inferences, towards a set of interested subscribers. It can thus be modelled as a publish/subscribe system or enterprise service bus. Several semantic publish/subscribe mechanisms have been proposed throughout the years. Early work was often based on RDF graph matching [17], [18]. Messages are represented using RDF graphs, while subscriptions take the form of graph patterns. More recently, Skovronski and Chio [19] presented an approach based on SPARQL protocol and RDF query language[3] (SPARQL) queries as subscriptions, which similarly perform matching based on RDF triples. The semantics that can be captured with graph matching algorithms or SPARQL queries are limited to property and hierarchical relationships. Li *et al.* [20] proposed a more expressive approach, using standard semantic reasoners to determine a match between messages and subscriptions. Subscriptions take the form of defencse advanced research project agency (DARPA) agent markup language and ontology inference layer (DAML+OIL)[4] classes, while messages are represented by instances. If the reasoner infers that a message instance belongs to a subscription class, then the message satisfies the subscription. The filtering approach presented in our work is similar to that proposed by Li *et al.*. However, we use the more modern web ontology language (OWL), instead of DAML+OIL. Our proposed context disseminator additionally supports subscriptions to be defined in the form of semantic web rule language (SWRL)[5] and Jena[6] rules, which further increases expressiveness through a wide range of built-ins, including comparison, string and mathematical operators.

In the context of web services, several semantic service matchmaking methods have been presented. OWLS-MX is a hybrid semantic web service matchmaker for OWL-S services [21]. OWL-S[7] is an ontology built on top of OWL for describing semantic web services. However, OWLS-MX only uses information about inputs and outputs, not preconditions and effects. For AEs, it is necessary to estimate the influence of services on their managed environment and as such an approach that incorporates preconditions and effects is necessary. More recently, several semantic matchmaking algorithms that do take into account preconditions and effects have been proposed, based on description logics [22], SPARQL [23], and SWRL [24]. In line with Bener *et al.* [24], we propose a semantic matchmaking algorithm based on SWRL. However, our algorithm takes into account a more complete definition of SWRL-atom subsumption relationships, thus resulting in more accurate matchings. Additionally, our algorithm, in contrast to the state of the art, supports the use of SWRL variables, which provide a means to semantically link inputs and outputs to preconditions and effects.

III. HIERARCHICAL CONTEXT DISSEMINATION

In this section, we identify and describe the interactions that take place between AEs in a highly dynamic and distributed management environment. Specifically, the section focusses on context dissemination between management components throughout a management hierarchy.

A. Architectural Overview

A modern network domain consists of a large number of physical (e.g., servers, switches, and network links) and logical (e.g., virtual machines, operating systems, software libraries, and services) resources. Fig. 1 depicts how a set of AEs interacts with such an infrastructure. The AEs within a management domain, are structured in a hierarchical fashion. In previous work, we have highlighted the qualitative and quantitative advantages of AE hierarchies, both in a generic network management context [7] and specifically for the management of clouds [25]. Throughout this article, a hierarchy with three main levels is used. The bottom level directly manages the physical resources, the middle level a cluster of resources, and the top level the entire domain. No assumptions are made concerning the number of AEs within a level. For example, a cluster of resources could be managed by a single AE, or by a hierarchy of collaborating AEs. The overall structure of the AE hierarchy, and more specifically the total number of sub-levels within each of the three main levels, obviously has its implications on scalability. Section VI studies this in more detail. Additionally, no assumptions are made concerning the physical location of the

[1] http://www.itil-officialsite.com
[2] http://tmforum.org/BusinessProcessFramework/1647/home.html
[3] http://www.w3.org/TR/rdf-sparql-query/
[4] http://www.daml.org
[5] http://www.w3.org/Submission/SWRL/
[6] http://jena.sourceforge.net/
[7] http://www.w3.org/Submission/OWL-S/

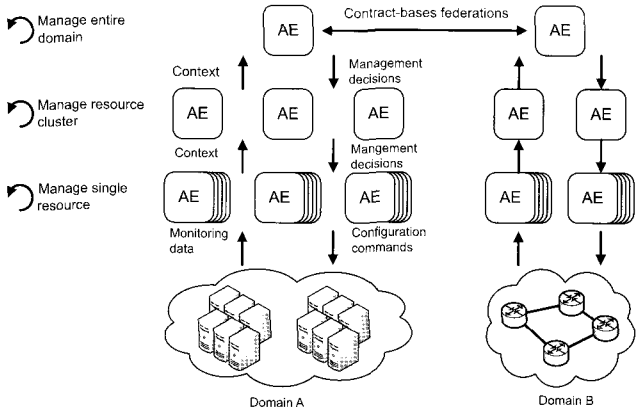

Fig. 1. An overview of the collaborative, AE-driven, and hierarchical management architecture.

AEs. They can, for example, be co-located with the managed infrastructure, executing them on the computing resources within the domain. Alternatively, to increase security and availability, part of the physical infrastructure can be reserved specifically for the AEs, effectively decoupling the managed environment from the management framework.

B. Resource Management

At the bottom level, AEs are directly responsible for the management and configuration of a single, or small group of, resources. For example, an AE managing a router could be, among others, responsible for configuring and managing a context-aware admission control function. They perform low-level maintenance tasks, such as monitoring, basic problem detection, and configuration. The AEs have three distinct responsibilities. Their main responsibility is to convert the continuous stream of monitoring data into semantically annotated context and forward an aggregated and summarized version to their parent AE. Second, the AE can be fitted with some basic reasoning functionality, allowing it to solve low-level problems, for which no global knowledge is necessary. For example, the AE could be responsible for managing and configuring an admission control component that rejects service requests if a managed server's load becomes too high. The third and final responsibility of the bottom level AEs, is the configuration of the network's resources. They offer a set of management functions for configuring the underlying resources. The implementation of these functions is vendor specific. The AEs should thus be aware of the type of resources they manage and should be able to translate generic commands into vendor specific ones. AEs at higher levels in the hierarchy can influence the managed environment by dynamically discovering and initiating the management functions of bottom level AEs.

C. Cluster Management

As we move up through the hierarchy, AEs take on more responsibilities and become capable of performing higher level and more widespread management tasks. For example, an AE managing a group of interconnected switches and routers is capable of changing domain-wide routing policies, such as configuring consistent quality of service (QoS) parameters across all devices. AEs at the second level are responsible for managing a cluster of resources. They model the state of their environment using context gathered through their child AEs. Depending on the current state of the environment, the AE can adapt the granularity and dissemination interval of the context it requests from its children. For example, when the underlying environment is in a stable state, only aggregated statistics are necessary in order to detect possible future problems. However, once a potential problem has been detected, more detailed and frequent context is needed in order to react in a timely fashion. This dynamic context dissemination process is further described in Section IV. At the middle level, AEs have a view over potentially many resources. Therefore, they are capable of detecting more intricate problems and deploying more widespread solutions.

D. Domain Management

At the top of the AE hierarchy is one, or a set of, root AEs. They have a high-level, highly aggregated view of the network. Their responsibilities include high-level management tasks that impact the entire domain. For example, in a cloud scenario this entails resource provisioning and service migrations across data centers. Therefore, they are capable of taking into account network related parameters and metrics in their decision making process. For example, services could be migrated closer to the customers that consume them, in order to optimize network-related parameters.

E. Federated Management

In addition to managing the high-level aspects of the domain, root level AEs are responsible for setting up loosely-coupled federations. They communicate with root AEs of other domains in order to negotiate contracts and set up federations. Such federations support novel types of collaboration, for example, allowing resource and revenue sharing. For example, in a cloud scenario, AEs could decide to migrate services from one cloud provider to another. This allows customer demands to remain satisfied even when an entire cloud provider's infrastructure becomes overloaded. To achieve this level of collaboration, providers need to exchange context information about availability, infrastructure, and resources. However, making detailed information available may not be possible due to scalability reasons, or wanted due to business reasons. The context filtering and aggregation techniques applied within a management domain, can therefore also be used in the context dissemination process between domains.

F. Context Continuum

The management responsibilities change throughout the AE hierarchy. This is reflected in differing capabilities and context requirements. Additionally, these responsibilities map to the policy continuum [26]. At the top, policies represent high-level business objectives and context is highly aggregated and summarized. As we move down the hierarchy, policies are gradually translated into low-level device configurations. A similar translation can be seen on the context level: Context is propagated upwards and is dynamically transformed, aggregated, and summarized. This allows us to maintain scalability throughout the hierarchy. At the bottom, AEs have a very detailed, yet narrow

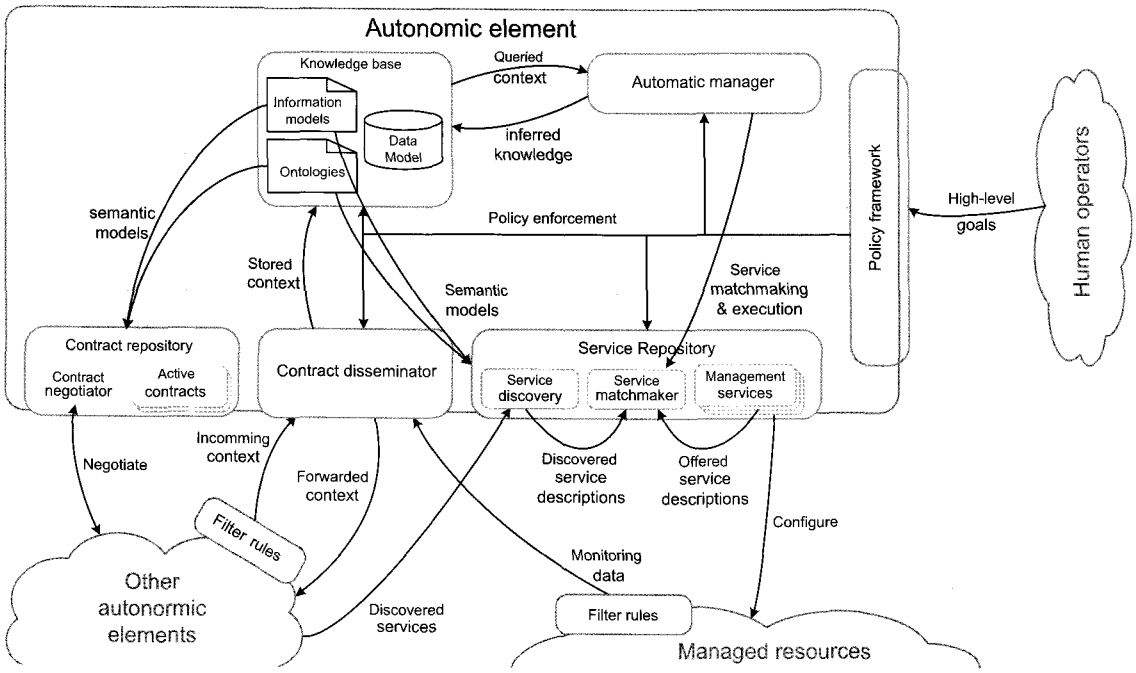

Fig. 2. Overview of the internal AE architecture and its interactions with other AEs and managed resources.

view of the domain, while at the top the view is domain-wide, but much less detailed. A context continuum can thus be defined in line with the policy continuum. Furthermore, policies at the highest level of the continuum will typically relate to context at the highest level and vice versa.

IV. AUTONOMIC ELEMENT ARCHITECTURE

The previous section focussed on the global system architecture and high-level interactions between AEs. In contrast, this section zooms in on the internal workings of the AE architecture and addresses the functional details of AE interactions. Section V further applies the algorithms and concepts introduced in this and the previous section to a practical resource allocation use-case in a federated cloud computing environment.

An overview of the internal AE architecture is depicted in Fig. 2. As illustrated, an AE consists of loosely coupled components that asynchronously interact with each other. The AE is made up of six main components: Knowledge base, context disseminator, autonomic manager, service repository, contract repository, and policy framework. Together, they form dynamic control loops, which adapt their behaviour based on the AEs location within the hierarchy and the current state of the environment. The remainder of this section discusses these components in more detail.

A. Knowledge Base

To perform their management tasks, AEs require information about their managed environment to reason upon and deduce new configurations. This information takes the form of monitoring data (received by the managed resources), remote context (received by the neighbouring AEs), and knowledge (locally deduced by the AE itself). New knowledge is typically deduced by the autonomic manager, through ontological reasoning or machine learning algorithms. For example, based on measured monitoring data, the AE could infer that a specific server hardware component has failed. The proposed AE architecture takes a dual approach in representing this knowledge. The bulk of the knowledge is stored in a data model. The AE is capable of interpreting, and reasoning upon, this stored knowledge using a set of embedded information models and ontologies.

A.1 Information Models and Ontologies

An information model represents the concepts, relationships, constraints, rules, and operations that define a specific domain. For example, in a cloud environment, an information model typically defines concepts that are of interest to the overall managed environment. Such concepts range from abstract and high-level (e.g., a server) to low-level (e.g., the load of a CPU core on a server). This enables different entities to be related to each other. The semantic capabilities of AEs stem from the information models and ontologies embedded within. They are used by other components inside the AE for reasoning and deducing new knowledge, which can in turn be added to the ontologies or stored in the data model. Although the information models and ontologies provide a consistent view of the management domain, their size can quickly expand up to hundreds of concepts and thousands of relationships. The performance of ontological reasoning is known to scale exponentially [27]. As such, every component only uses the subset of the complete information models and ontologies that are relevant to its own context and environment, which significantly reduces overhead. The components must define the parts they are interested in.

A.2 Data Model

All data relevant to the AE is stored in the data model. The data model is an instantiation of the information models. It allows storing values and instances for the defined concepts, and

thus acts as a database that characterizes the state of the managed resources as well as other business oriented aspects, such as high-level policies.

B. Context Disseminator

The data being stored into the data model can originate locally (e.g., because new knowledge has been deduced) or remotely (i.e., from other AEs or managed resources). When context originates remotely, a decision needs to be made on which context needs to be transferred from other AEs and managed resources to the local AE. As discussed in [7], forwarding all context to all other AEs and managed resources is simply not possible for scalability reasons: The filtering of context can reduce the contextual overhead considerably. In the proposed architecture, this is the responsibility of the context dissemination component. For this, the context disseminator uses the publish/subscribe paradigm. In this paradigm, context consumers express their interest in data and/or information that is meaningful to them. In our proposed architecture, AEs specify interest in information by way of filter rules, which identify the type of information or context in which they are interested. Depending on the type of interaction, several types of filter rules are supported. For example, in a typical network management scenario, simple network management protocol (SNMP) traps [28] can be used to request monitoring information from a router. Additionally, semantic filter rules are supported for the exchange of context between AEs. This allows them to request context based on its actual meaning. In this approach, context messages take the form of ontological instances, while filter rules can be specified as ontological concepts or semantic rules (e.g., SWRL or Jena). Ontological reasoners are then used to match this semantically defined context with semantic filter rules. As this article focusses on the architectural components and the interactions between them, the algorithmic details of the filter rule matching process are omitted. The reader is referred to our previous work for more details [29]. Nevertheless, some examples, specific for the management of a large-scale cloud computing data center, are given in Section V.

The contextual requirements of an AE change with the state of its managed environment. Existing approaches often disseminate all context that might be needed at some point at all times. In contrast, we propose a dynamic context dissemination process, where filter rules are automatically generated, based on the current state of the managed environment. For example, a reasoning component that migrates applications when a server becomes overloaded can choose to only monitor the average load of the server as a whole. When this average load indicates that the server is overloaded, the reasoning component then requests additional information, such as the footprint of every application individually. The context dissemination component supports the context-sensitive generation of filter rules through ontological reasoning. The conditional contextual requirements of each component of an AE are defined by the context dissemination component. Among others, dependencies between contextual requirements can be introduced (e.g., stating that certain context may only be requested if other context has a particular value). For example, detailed statistics about the processes running on a server might not interest the AE managing this server, unless its aggregated resource load becomes too high. For more details on this generation algorithm, the reader is referred to [30].

C. Autonomic Manager

The autonomic manager is responsible for detecting and solving problems and sub-optimal performance. It performs this complex task through the use of specialized semantic reasoners, learning algorithms and management algorithms. Depending on the goals of the AE, different algorithms and reasoners, of differing complexity, may be active within the same autonomic manager. The autonomic manager is the premier consumer and producer of contextual data: It continuously extracts context from the knowledge base and stores newly inferred knowledge into it. To make effective decisions, it also requires contextual data that is only available remotely. As such, it depends heavily on the context dissemination process. This context is used to assess the current state of the environment and plan corrective actions. As such, this component forms the heart of the control loops of the AE. Depending on the scope and complexity of the management problem, as well as the location of the AE in the hierarchy, different management algorithms may need to be used within the control loops. This results in a wide variety of dynamic control loops, with varying responsibilities.

D. Service Repository

When the state of the managed environment no longer matches the desired state, the AE's autonomic manager constructs a plan containing corrective actions that need to be performed. However, these actions need to be mapped to the available management services. Additionally, AEs offer specialized functionality based on their location within the hierarchy and environment. Therefore, they might not be able to perform all planned actions themselves. The service discovery component is responsible for both mapping actions to services (i.e., matchmaking), as well as discovering functionality offered by other AEs.

The management services come in two forms. The first type is capable of performing well-defined and straightforward tasks that alter or configure the managed resources. In a cloud environment, typical examples of such management services are the migration of virtual machines, the allocation of resources, and the hibernation of cloud servers. The second type of management service can be used to steer the autonomic manager. For example, in a cloud computing scenario, the top level AE might decide it is necessary to migrate a hosted cloud service from one data center to another. However, as it does not have a detailed view of the target data center, it could instruct the AE responsible for this data center to find a suitable server (or set of servers) for hosting the service. The AE responsible for this data center should then have a management service that instructs its autonomic manager to allocate resources for a specific hosted service.

Services are described through a set of inputs, outputs, preconditions, and effects (IOPEs). The inputs represent the context that is required by the management service in order to perform its task. On the other hand, the outputs depict context that will be generated by the management service. The preconditions

define the environmental conditions that must hold for the service to be usable, while the effects describe the conditions that will hold after the service's execution. Together, the preconditions and effects thus describe how the managed environment will change when the management service is executed. This allows AEs to better match these services with their planned actions. Additionally, variables may be attached to specific IOPEs, which allows preconditions and effects to be semantically linked to each other and inputs or outputs. The proposed matchmaking algorithm uses the subsumption relationships of IOPEs to determine compatibility between service descriptions and goals.

E. Contract Repository

In a federated cloud computing scenario, the interactions between AEs within different management domains need to be governed by a contract negotiation process, as new issues, such as trust and conflicting management policies, arise that are not present within a single management domain. As previously stated, these contracts need to be augmented with semantics, based upon a shared information model, in order to support understanding and correct interpretation between the participants of the federation. Contracts are similar to service descriptions, as IOPE definitions can be used to semantically describe costs and benefits of the involved parties. While service descriptions only describe functional properties, contracts may also include business-related aspects of the interactions.

F. Policy Framework

Autonomic environments are typically governed by policies. These policies represent the high and low-level goals human operators can introduce into the AE, used to steer the management process. The policy framework is the entity that collects these policies for further forwarding to the specific components inside the AE. In the policy framework, policy related tasks such as the detection of conflicts between policies are maintained. Several policy conflict detection algorithms have been proposed in previous work [31], [32]. As policies form an inherent part of the inner workings of an AE, the policy framework interacts with all components inside the AE. For example, in the context dissemination, the filter rule generation process in the context dissemination can be governed by a policy that limits the amount of context that can be requested from a particular AE. The effect of such a policy on the filter rule generation process is that the generated filter rules will either request context at a lower frequency or increase the level of performed aggregation of lower level context.

As previously stated, policies take different forms throughout the AE hierarchy. This corresponding policy hierarchy is also referred to as the policy continuum. In order to achieve true autonomic behaviour, and thus further reduce management complexity for human operators, algorithms are needed that automatically perform the translation of policies throughout the hierarchy.

V. MANAGING THE CLOUD

This section applies the generic concepts introduced in Sections III and IV to the specific area of cloud computing. As previously stated, the size of cloud computing data centers is growing steadily. In turn, this is causing a significant and ever-increasing growth in context information that needs to be processed by management components. Consequently, our presented framework greatly increases efficiency and scalability of cloud management architectures by intelligently disseminating and dynamically filtering context based on the current state of the underlying resources. The contributions of this section are threefold. First, it is shown how the AE management hierarchy is mapped to cloud data centers. Second, the context requirements at all levels of the hierarchy are identified. Third, it is described how the presented framework is capable of meeting these dynamic context requirements in a scalable manner.

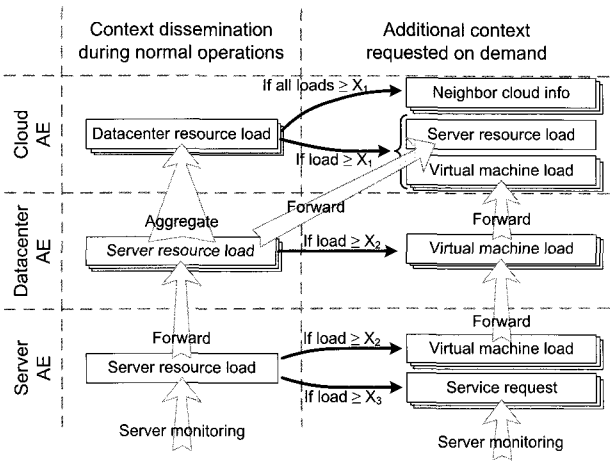

Fig. 3. The context continuum for resource allocation in cloud computing; context is forwarded and aggregated through the AE hierarchy using dynamic context-aware filter rules.

An important aspect of managing a cloud computing infrastructure is the allocation of physical server and network resources to virtual machines and hosted services. In previous work, we showed how existing resource allocation algorithms can be adapted for use in an AE hierarchy [25]. In contrast, this article focusses more on the interactions between AEs in order to facilitate such hierarchical collaborations. As such, specific algorithmic details are omitted.

When performing resource allocation in a cloud computing environment, several different types of context are needed at different levels of the AE hierarchy depending on the current environment. AEs dynamically adapt their filter rules in order to receive the correct context based on the current situation. Additionally, context is aggregated in order to increase scalability. Fig. 3 shows the context requirements for the resource allocation case described in this section. As the performed management tasks significantly differ at the different levels in the hierarchy, this section is split into three subsections, which present AE operations at different levels in the hierarchy. For each hierarchical level we discuss the details of the context dissemination, management services and autonomic manager components.

Throughout this section several examples of filter rules and context dependencies are given. The examples use the OWL Manchester syntax[8] and SWRL rules. The concepts used within

[8]http://www.w3.org/TR/owl2-manchester-syntax/

these rules are derived from the ontologies within the knowledge base. This allows the language in which these rules are defined to be dynamically expanded, by extending the ontologies. The ontologies used for the presented examples are described in our previous work [16].

A. Cloud Management

The top level AE has several important responsibilities. First, it orchestrates the interactions across data centers. Second, it negotiates, configures, maintains and monitors federations with other cloud providers.

A.1 Context Dissemination

The root AE is concerned with resource allocation on a data center level. To determine the need for virtual machine migrations across data centers, it continuously requests statistics about average, maximum and minimum CPU, and memory consumption from its child AEs, which operate on the data center level. Obviously, to maintain scalability, the statistics are aggregated over all servers within a data center. This means that under normal conditions, the root AE only maintains a few context values per data center. One of the filter rules associated with this context requirement is defined in the OWL Manchester syntax as follows:

Message **and** hasPayload **some**
(AggregatedMeasurable
and hasEntity **some** Datacenter
and hasType **value** CentralProcessingUnit)

The rule states that the AE is interested in messages that contain context about aggregated measurable values related to data centers. Additionally, the type should be a CPU. A similar filter rule can be defined for the type *MemoryModule*, which would pick up memory-related context.

If one of the average resource values of a data center goes over a pre-specified threshold X_1, a new set of filter rules is dynamically generated based on the change in context, at which the root AE starts requesting more detailed information. More specifically, it instantiates filter rules that request detailed statistics about servers with a load higher than X_2. These statistics include aggregated resource loads of the server itself, but also information of its hosted virtual machines. The filter rule that admits measurements of overloaded resources is the following:

Message **and** hasPayload **some**
(AggregatedMeasurable
and hasEntity **some** CloudServer
and hasValue **some** double[$> X_2$])

The automatic context sensitive generation of filter rules is a task of the context dissemination component. For this, dependencies are introduced between the different context types at different layers in the context continuum. A context dependency from context type A to context type B denotes that A can only be requested if B has a particular value, defined in the dependency. For example, to allow the generation of the above example, the following context dependency is used:

$$\begin{aligned} &\text{HighDatacenterLoad}(?\text{load}) \\ &\wedge\ \text{hasContextDependencyList}(?\text{load},?\text{list}) \\ &\wedge\ \text{hasContextDependency}(?\text{list},?\text{dep}) \\ &\wedge\ \text{hasContextType}(?\text{dep},?\text{max}) \\ &\wedge\ \text{TooHighMaxServerLoad}(?\text{max}) \\ &\Rightarrow\ \text{ContextToQuery}(?\text{load}) \end{aligned} \quad (1)$$

The concepts HighDatacenterLoad and HighServerLoad are also part of the context dependency and are defined as follows:

HighDataCenterLoad ≡
AggregatedMeasurable
and hasEntity **some** Datacenter
and hasType **value** CentralProcessingUnit
and hasValue **some** double[$> X_1$]

and

HighServerLoad ≡
AggregatedMeasurable
and hasEntity **some** CloudServer
and hasType **value** CentralProcessingUnit
and hasValue **some** double[$> X_2$]

As can be seen, it is in these definitions that the two thresholds, X_1 and X_2, are defined. In this case, the generation of filter rules works as follows. At the data center's AE, each time the aggregated data center load is updated, the context dependency is checked, which may lead to the context type HighServerLoad being classified as ContextToQuery. This leads to the generation of a filter rule requesting HighServerLoad instances. These HighServerLoad instances are then classified at the server's AE, according to the above definition.

The second task of the root AE is to configure and manage federations with other cloud providers. If the average load of all data centers within the cloud becomes too high, the root AE can no longer migrate virtual machines between data centers. The problem can then only be solved by leasing infrastructure from another cloud provider and setting up a cross-domain collaboration. This will trigger several filter rules for gathering the context needed in order to perform the resource re-allocation and negotiate a contract. This includes context about the virtual machines that can be outsourced and billing and availability information of other cloud provider networks.

A.2 Autonomic Manager

As shown, the context dissemination process dynamically adapts itself to changes in the environment. When the autonomic manager needs to perform resource migrations across data centers or cloud infrastructures, the filter rule generation algorithm already made sure that context necessary for executing the resource allocation algorithms is present in the AE's data model. The autonomic manager uses the available context to determine which virtual machine(s) to migrate and select a suitable destination. However, the root AE does not have a detailed view on the destination data center or cloud infrastructure. Therefore, it contacts the AE responsible for the destination through a suitable management service. This management service triggers the

destination AE's autonomic manager, which contains a suitable algorithm, and the required context, to determine to which exact server to migrate the virtual machine(s). The root AE thus governs the migration process, but delegates the actual low-level configuration tasks to its child AEs.

A.3 Management Services

The root AE offers a few management services, which can be used by its own autonomic manager or in the contract negotation process when interacting with other root AEs. First, it offers a composite management service for migrating virtual machines from one data center to another. The service is actually composed of several lower-level management functions. It stops the virtual machine on its source server, releases the reserved resources and contacts the destination data center AE to find a suitable destination server. Second, it contains some management services for setting up federations. For example, it could offer a service that allows other cloud providers to lease part of their infrastructure. These services are used as a basis for the contract negotiation process.

B. Data Center Management

At the data center level, AEs are responsible for managing a single data center. However, as the size of a data center grows, scalability can be maintained by allowing several AEs, grouped in a hierarchy, to each manage part of the data center.

B.1 Context Dissemination

Under normal conditions, the data center AE periodically queries its children, for context concerning the aggregated load of their managed server(s). It thus maintains a few context values per server or sub-cluster. This context is needed for several reasons. First, it needs this information to satisfy the context requirements of the root AE, which needs aggregated statistics about the data center resource load. The data center AEs thus use the gathered server- and cluster-granularity statistics to infer data center-granularity context. Second, if the load of one of its servers surpasses the threshold X_2, it needs to request more detailed context of that server in order to prepare for a virtual machine migration. Once the threshold is reached for a server, the data center AE adapts the filter rules of the corresponding server AE in order to request context about the virtual machines of the server.

As stated, when context is passed between levels in the management hierarchy, it is dynamically aggregated and transformed. For basic types of aggregation (i.e., averaging or maximizing values such as the aggregated statistics of the load), the filter rules themselves can be used to perform such transformations. As discussed in [30], the context model defines operators that allow aggregating the data. The type of aggregation operators supported correspond with those that are possible in a typical database query language such as SQL. For more advanced aggregation (e.g., taking 2nd-order statistics of data), the context should be aggregated by the autonomic manager itself. In this case, the aggregated context is regarded as a new piece of inferred knowledge. Note that the aggregation obviously has an impact on the precision of context at the root of the tree: The more information is aggregated, the less detailed the view a parent AE has on the status of the underlying network. Introducing an additional layer into the hierarchy can therefore also result in a loss of precision if a high amount of aggregation is performed. However, in this case, the context dissemination framework allows requesting more detailed context based on the received aggregated value. As such, the loss of precision can be controlled by carefully specifying contextual requirements.

B.2 Autonomic Manager

It features a reactive management process that corrects the overload of cloud servers by migrating virtual machines from one cloud server to another. Existing resource allocation algorithms can be employed to decide how the migration should be performed [33]. Similar to the root level, the adaptive filter rule generation algorithm makes sure that the necessary context is available when the resource allocation algorithm is triggered.

B.3 Management Services

As stated, one of the responsibilities at the data center level is virtual machine migration. As such, a management function is present to perform such migrations. This function is actually a composition of several low level management functions. More specifically, it consists of the following steps. First, the virtual machine needs to be stopped on the original cloud server and its resource reservations need to be cancelled. Second, the virtual machine needs to be started on the newly selected cloud server and the necessary resource reservations need to be made.

Additionally, the data center AE offers a management service that interfaces with its autonomic manager. More specifically, the root AE needs to be able to trigger the resource allocation process when it migrates a virtual machine from one data center to another. The root AE only determines to which data center a virtual machine should be moved. The data center AE can itself decide which cloud server or servers to use. This management service takes as input a virtual machine and as effect places that virtual machine on one of its servers. Internally, it triggers the data center AE's autonomic manager which uses a resource allocation algorithm to find a suitable server for the virtual machine.

C. Server Management

The bottom level AEs are each responsible for a single cloud server. Their main goal is to closely monitor this server's state and configure it through a set of management services.

C.1 Context Dissemination

The server AE directly interfaces with the underlying managed resources. As such, it must be able to communicate with them in a device specific manner. Here we assume that the cloud servers can be queried via the SNMP protocol. In contrast to higher levels, filter rules take the form of SNMP traps instead of semantic definitions. Normally, the server AE periodically queries the CPU and memory load of its underlying server using SNMP GET, which is a pull-based mechanism. Additionally, it sets an SNMP trap that triggers when one of the server's resources reaches the threshold X_2 or X_3 (with $X_2 \leq X_3$). When the threshold X_2 is reached, the server AE starts requesting additional information about the individual virtual machine

Table 1. A service description of a management function that allows AEs to allocate physical resources to a virtual machine.

IOPEs	SWRL atoms
Inputs	VirtualMachine($?v$), QuantifiableResource($?r$)
	xsd:long($?i$)
Outputs	
Preconditions	CloudServer($?s$), executes($?s$, $?v$)
	hasState(?s, TurnedOn), consistsOf($?s$, $?r$)
	hasCurrentValue($?r$, $?l$), hasMaximum(?c, $?m$)
	swrlb:substract($?a$, $?m$, $?l$)
	swrlb:greaterThanOrEqual($?a$, $?i$)
Effects	hasReservedResource($?v$, $?t$), hasEntity($?t$, $?r$)
	hasValue($?t$, $?i$)

resource requirements. Note that this change in filter rules is not directly initiated by the server AE, but rather a consequence of a request for more detailed virtual machine context by the data center AE. Finally, if one of the server's resources reaches X_3, a problem has occurred that cannot be solved swiftly enough at the higher levels. This initiates an admission control mechanism. Additional filter rules that intercept detailed context about individual service requests, which is needed by the admission control component, are instantiated.

C.2 Autonomic Manager

The server AE has a very narrow view on the managed environment. Therefore, it can only offer basic solutions to occurring problems. Its main responsibility is initiating the admission control algorithms when the resource load of the server passes the threshold X_1. When this happens, all service requests arriving on the server are placed in a queue. Through suitable filter rules, the server AE is notified whenever a new request arrives in the queue. The admission control algorithms decide whether or not to let the request through and notify the server via a suitable management service. Note that admission control is only a last resort. Whenever possible, an overloaded server problem will be efficiently solved at a higher level and admission control will not be necessary.

C.3 Management Services

The server AE forms the gateway to the underlying managed resources. Therefore, it offers a plethora of management functionality that can be used by its own autonomic manager and its parent AE to influence the state of the environment. The offered management services include: Changing a server's state, change the resource allocation of a virtual machine, start or stop virtual machines, and turn on or off the admission control system. Table 1 depicts an example service description modelled using SWRL atoms. It represents a management function that allocates a specified amount of resources, of a specific hardware component to a specific virtual machine. The service takes three inputs: A reference to a virtual machine $?v$, the specific resource $?r$ that should be reserved (e.g., a CPU core or memory module) and the amount $?i$ of the resource that should be reserved. The preconditions are somewhat more complex. The first three state that the server $?s$ that hosts the virtual machine $?v$ should be turned on. The remaining atoms calculate the currently available amount of resources of $?r$ and store it in variable $?a$, Finally, the last precondition states that there should be at least $?i$ resources available on $?r$. The effects state that after successful execution of the service, the requested amount of resources will be reserved for $?v$ on $?r$. Note that by way of SWRL variables, inputs and outputs can be linked to preconditions and effects. For example, the input variable $?v$ is reused in the preconditions and effects. Consequently, the semantic matchmaking algorithms and reasoners know these conditions refer to the same virtual machine as specified in the inputs.

VI. EVALUATION

The context dissemination process plays a central role in the AE's autonomic control loops. Therefore, its performance and scalability severely impact the management system in general. This section presents an evaluation of the proposed context dissemination framework. First, an overview of our prototype implementation is given. Second, the scaling behaviour of the presented hierarchical approach is studied using analytical formulations. Third, the impact on performance of introducing semantics into the context dissemination process is evaluated using the presented prototype. The evaluation uses the cloud management scenario detailed in Section V. The contextual requirements, and context dissemination process detailed there are used throughout this section.

A. *Prototype Implementation*

A prototype AE implementation, based on the presented concepts, was created. The prototype was built in Java, based on the Pellet[9] OWL 2 reasoner version 2.1.1 and OWL-application programming interface (OWL-API)[10] version 3.0.0. In addition to OWL 2 reasoning, Pellet supports DL-safe SWRL rules [34]. Jena rules are supported through Jena's own built-in rule reasoner, of which version 2.6.2 was used. The embedded information models take the form of OWL ontologies, using the directory enable networks-ng (DEN-ng) ontology (DENON-ng) as a basis [35]. DENON-ng is an ontology derived from the DEN-ng information model [36]. Additionally, DENON-ng was extended with a model for representing cloud computing specific concepts. A detailed description of these ontologies can be found in our previous work [16]. The Jena TBD high-performance triple store is used as a data model. The context disseminator implementation supports several types of filter rules. They can take the form of OWL 2 class definitions, SWRL rules, Jena rules or SNMP traps. The actual context messages are defined as OWL instances, and are matched with the semantic filter rules using the embedded Pellet and Jena reasoners. The filter rule generation process uses context dependencies defined in SWRL to generate filter rules of different types, using the embedded Pellet reasoner. Service descriptions are also defined using SWRL atoms. A custom service matchmaking algorithm, which uses the Pellet reasoner to determine subsumption relationships between these atoms, was devised and implemented. The autonomic manager supports different types of management algorithms, such as rule-based systems or neural

[9] http://clarkparsia.com/pellet

[10] http://owlapi.sourceforge.net/

networks. The policy framework and contract repository are currently not implemented within the prototype.

B. Context Dissemination Scalability

This section studies the amount of context that is received by AEs throughout the hierarchy and determines how the hierarchical structure can be exploited to improve scalability. At the lowest level of the hierarchy only a single server is managed by the AEs. As such, the amount of generated context depends on factors such as the number of hosted virtual machines and received service requests. As the number of virtual machines that can be hosted on a single physical server is limited, this is not expected to severely impact scalability. On the other hand, at the data center and cloud levels, the amount of context depends on the number of servers within a data center and the number of data centers, respectively. When the data center is managed by a single AE, it eventually becomes overloaded with context as the number of servers within grows. Therefore, a large data center should be managed by multiple AEs, which are also structured in a hierarchy.

This section presents an analytical model that calculates the amount of context that needs to be processed at different levels within the data center layer. The cloud level model can be similarly defined, but is omitted for brevity. Subsequently, the model is employed to show how scalability of the context dissemination process can be maintained by using hierarchies of AEs. The model is based on the resource allocation use case presented in Section V and the context requirements defined in Fig. 3.

B.1 Analytical Formulation

Consider a data center with S servers that host, on average, V virtual machines each. Additionally, R different resources are monitored (e.g., CPU or memory). The data center AE hierarchy consists of n levels, with level 1 the lowest and n the highest data center level. Level 0 thus represents the server level and $n+1$ the lowest cloud level. T_i is defined as the total number of AEs at level i. As such, $T_0 = S$, as every server is managed by a unique AE. C_i is defined as the number of child AEs governed by every AE at level i, while M_i is a configurable parameter that determines the maximum number of child AEs at level i. Based on the input variables S, V, R, n, and M_i, the amount of context that is processed by every AE can be calculated. The context calculation is based on the value of C_i, which in turn depends on T_i. Therefore, T_i is first defined:

$$T_i = \left\lceil \frac{T_{i-1}}{M_i} \right\rceil. \tag{2}$$

Plainly, the total number of AEs at level i equals the number of AEs at level $i-1$ divided by the maximum number of child AEs at level i, rounded up. Similarly, C_i is defined as follows:

$$C_i = \frac{T_{i-1}}{T_i}. \tag{3}$$

In other words, the number of child AEs of every level i AE equals the total number of AEs at level $i-1$ divided by the total number of AEs at level i.

Now, let N_i represent the total amount of context received by every AE at level i per time period during normal operations. The amount of context is measured using an abstract unit, which counts the pieces of information that are disseminated, such as a single server or virtual machine resource measurement. During normal operations, every AE receives a single aggregated measurement for each resource type every time period from each of its child AEs. N_i is thus calculated as follows:

$$N_i = RC_i. \tag{4}$$

When an overload is detected, additional context is generated. At the lowest data center level, the AEs are in direct contact with server AEs. Therefore, when the load of one of those servers exceeds X_2, it starts requesting one additional measurement for each resource type for each of the server's virtual machines, or a total of RV. On higher levels, additional context is only requested of servers with load higher than X_2, from child AEs whose managed cluster's aggregated average load exceeds X_1. However, here not only virtual machine resource information is requested but also of the server itself, or a total of $R + RV$. Now, let P equal the percentage of servers whose measured load exceeds X_2. The amount of additional generated context O_i at level i when a single child cluster becomes overloaded can then be calculated as follows:

$$O_i = \begin{cases} RV, & i = 1 \\ (R + RV)\, P \prod_{j=1}^{i-1} C_j, & i > 1. \end{cases} \tag{5}$$

In the second case, the total number of servers within a child cluster is calculated by $\prod_{j=1}^{i-1} C_j$. The percentage of overloaded servers within a child cluster therefore equals $P \prod_{j=1}^{i-1} C_j$.

B.2 Evaluation

To evaluate the proposed hierarchical model, the parameters, introduced in the analytical model, need to be quantified. We assume $R = 2$, as two resources, CPU and memory, are measured. The average number of virtual machines adds only a constant factor to the results and its value therefore does not matter. For reference, a value $V = 10$ was arbitrarily chosen. The percentage of overloaded servers was set to $P = 25\%$. The evaluated data center consists of up to $100,000$ servers.

Fig. 4 shows the amount of received context per AE for a two level data center hierarchy, with $M_1 = 250$. Fig. 4(a) depicts received context under normal operations, when no child cluster of the AEs is overloaded. When a child cluster of an AE becomes overloaded, additional context is generated. Fig. 4(b) shows the amount of additional generated context per overloaded child cluster. Under normal circumstances, the amount of generated context on the lowest level is obviously limited, as there is a maximum limit of 250 server AE children, which limits the received context to $RS = 2250 = 500$ units. In contrast, the second level contains only a single AE, which governs all level 1 AEs within the data center layer. Consequently, its received context increases linearly with the size of the data center, which clearly presents a scalability bottleneck once the data center grows too large.

The depicted results are only valid when context filtering takes place. On the other hand, if all context is propagated upwards, the amount of context received by every AE is significantly higher. For example, in a data center with $100,000$

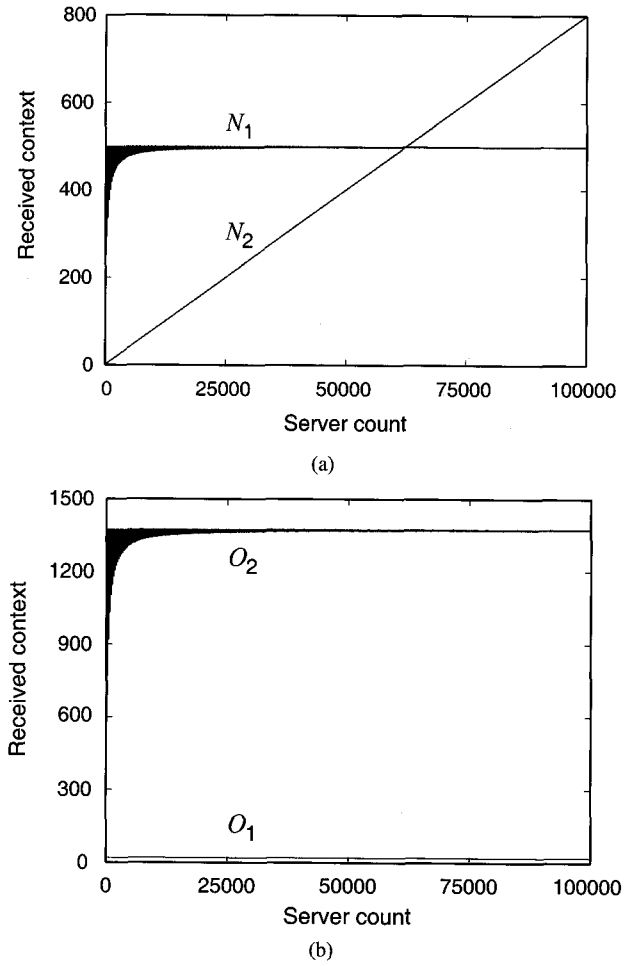

Fig. 4. The amount of context received by data center AEs for a 2 level data center hierarchy: (a) During normal operations and (b) for an overloaded cluster.

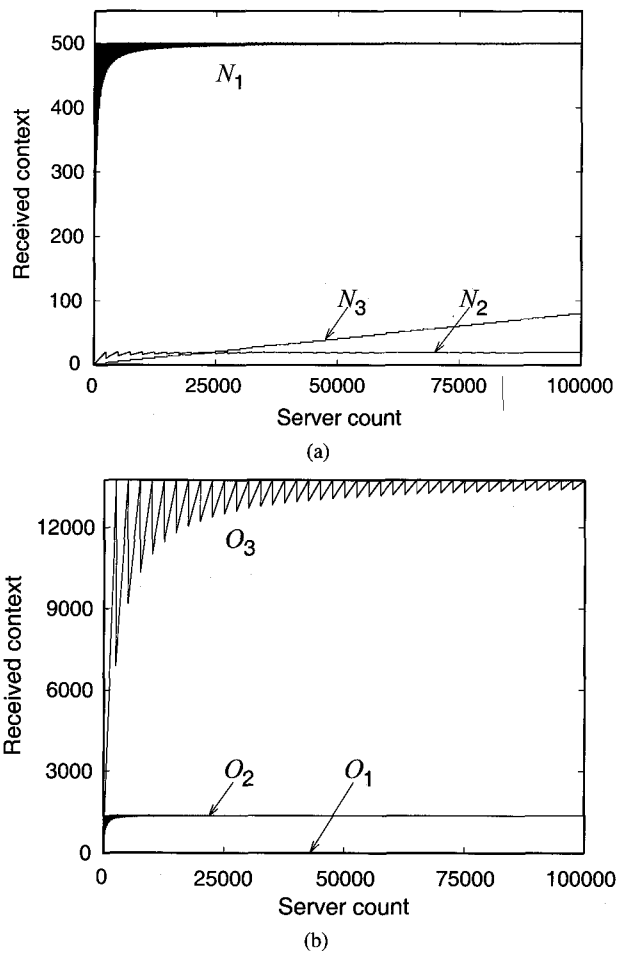

Fig. 5. The amount of context received by data center AEs for a 3 level data center hierarchy: (a) During normal operations and (b) for an overloaded cluster.

servers, AEs on the first data center level would have to process 5500 instead of 500 units of context. Even worse, the single AE on the second data center level, would need to process detailed context information about all servers at all times, which comes down to 2, 2 million units of context. This clearly illustrates the need for intelligent context filtering.

As shown in Fig. 4(b), the additional generated context when a cluster becomes overloaded has an upper limit. However, the generated context reaches that limit as soon as $s \geq M_1$. This results in a high context load even for relatively small networks. To solve this issue, only detailed context information about the subset of overloaded servers nearest their resource limit is propagated to the top of the hierarchy. At the top level, resource migrations is then performed to solve the overload of these servers. Recursively repeating this process down to the bottom of the management hierarchy for the other overloaded servers solves the entire overload, while circumventing the described bottleneck. For example, if, instead of detailed context about the 25% overloaded servers, only context about the 100 most overloaded servers is propagated upwards, O_2 is reduced from 1375 to 550.

As shown, the top data center level presents a scalability bottleneck in terms of generated context, as an ever-increasing number of servers is governed by a single top-level data center AE. However, this problem can be reduced by adding additional levels to the hierarchy. Fig. 5(a) shows how context overhead per AE is significantly reduced by introducing a third data center level into the hierarchy, with $M_1 = 250$ and $M_2 = 10$. The second data center level is now limited to 10 child AEs per AE, thus limiting the overhead generated under normal circumstances. Again, the number of children of the top level (in this case the third), is theoretically unlimited. Therefore, the scalability bottleneck shifts there. However, due to the introduction of an additional level, ten times as many servers are necessary to generate an additional child for the top level AE. Consequently, the received context increases much slower. Once the context received at the top level AE reaches a prespecified limit, a new level can be introduced, effectively allowing unlimited scalability of received context as a function of server count. The problem related to introducing additional hierarchical levels becomes apparent in Fig. 5(b). In this case, the top level AE's children are each responsible for up to 1000 servers, instead of 100 in the 2 level hierarchy. As such, the generated context when an overload occurs has also increased tenfold. Again, a significant reduction can be achieved by limiting the number of servers that are returned to only a fixed amount with the highest load. The resource allocation algorithm could start by migrating virtual machines hosted by these most overloaded servers. The underlying AEs could then themselves solve the problems within their

own sub-cluster.

In summary, we have shown how changing the structure of the AE hierarchy can be leveraged to effectively limit the amount of context that must be processed by every AE. More specifically, by increasing the number of levels in the AE hierarchy, the load on the top level AE can be significantly reduced, greatly increasing the scalability of the context dissemination process. As a downside, increasing the number levels in the hierarchy has been shown to slightly decrease optimality of the management algorithms [25].

C. Semantic Reasoning Overhead

Semantic reasoning using ontologies is computationally hard. Therefore, the reasoning steps of the context dissemination process are expected to impact performance. This section studies the impact on performance by semantic reasoning in the filter rule generation and context filtering steps. More specifically, the total delay introduced by the reasoning processes was measured, which allows us to determine whether or not semantic reasoning is feasible when managing large scale networks. As in the previous section, we focus on the data center level. Determining the reasoning time in a 100,000 server data center, with a two-level data center management hierarchy, where the lowest level AEs manage up to 250 server AEs and a single top level data center AE manages up to 400 low level data center AEs. In the presented results, Jena rules were used during the filtering process. This choice was made for two reasons. First, Jena and SWRL rules offer greater expressive power than pure OWL 2 class definitions. Second, unlike SWRL in Pellet, the Jena reasoner allows rules to be used without OWL inferencing. Therefore, Jena rules offer better performance for the price of decreased inferencing capabilities. All tests were performed on a server with two dual-core AMD Opteron 2 Ghz processors and 4 GB memory, running Debian 5.0 and Linux kernel 2.6.30.

Fig. 6 depicts the reasoning time for the filter rule generation process for both levels of the data center hierarchy. Filter rules are generated by the AE itself and then distributed to the child AEs. As such, the bottom data center AEs generate filter rules which are then sent to the server level AEs. The filtering itself is performed by the server AEs. Fig. 6(a) presents the reasoning time for generating filter rules for up to 250 server-level child AEs. The curve labelled N_1 shows the reasoning time under normal circumstances (i.e., when none of the servers are overloaded). In this case, the reasoning process takes less than 250 ms for up to 250 servers. In contrast, curve O_1 depicts the reasoning time for a growing number of overloaded servers and a total of 250 servers. Even when all servers within the cluster are overloaded, and consequently a huge amount of context is received, determining which filter rules to activate is feasible in less than 30 s. However, this is highly unlikely to occur, as the overload should be solved when it starts occurring and not after all servers have become overloaded. In case the overload is limited to a fraction of all servers, filter rule generation takes only a few seconds at most. Fig. 6(b) depicts the reasoning time for generating filter rules at the top data center level, for 400 child AEs and as a function of the number of overloaded child cluster. Every cluster is assumed to contain 250 children, of which 25 are overloaded for overloaded clusters. The figure shows that the

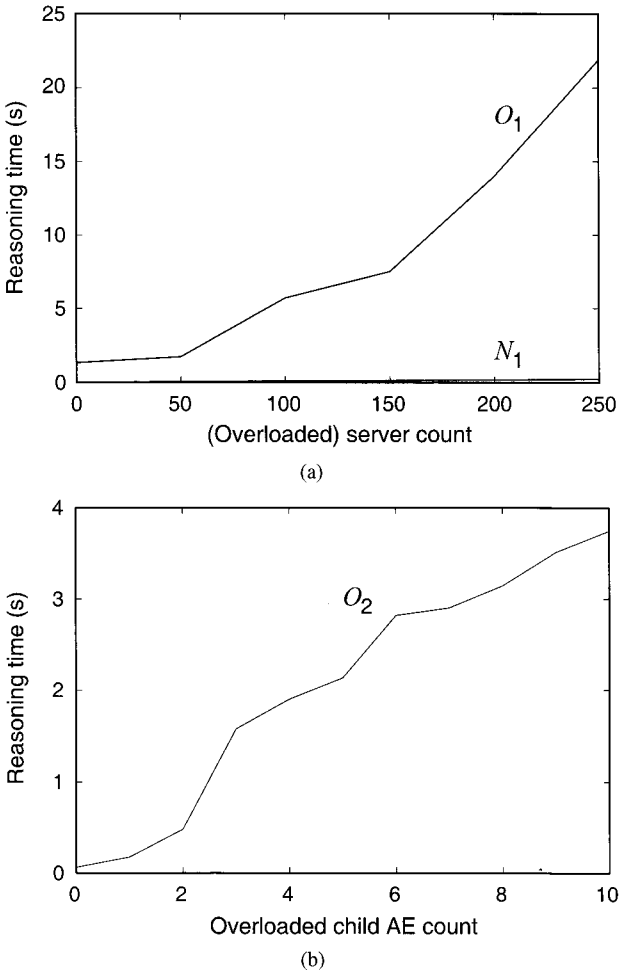

Fig. 6. The reasoning time for generating filter rules: (a) Bottom data center AE level and (b) top data center AE level.

reasoning time remains under 1s for up to 3 overloaded cluster, or 75 overloaded servers. For 10 overloaded clusters, and thus 250 overloaded servers, the filter rule generator is still capable of generating new filter rules in less than 4 s. The increasing reasoning time in both scenarios is due to the increasing amount of knowledge that is being stored into the ontological context model, thus complicating the reasoning process.

The reasoning time for the actual semantic filtering process is shown in Fig. 7. As stated, the generated filter rules are forwarded to the underlying child AEs, where the actual filtering takes place. The figure thus depicts results for filtering at the server level and the bottom data center level, which forward context to the bottom data center level and top data center level respectively. The server AE maintains a single server. Therefore, Fig. 7(a) depicts the reasoning time as a function of the number of virtual machines hosted on the server. The number of virtual machines directly influences the amount of generated context. In an overloaded scenario, with 20 virtual machines, it takes the context disseminator less than 60 ms to match the produced context with the defined filter rules. The reasoning time for the bottom data center AEs is shown in Fig. 7(b). At this level, AEs must process and disseminate detailed context of up to 250 servers. Under normal circumstances, only a few aggregated context values are disseminated. However, for every

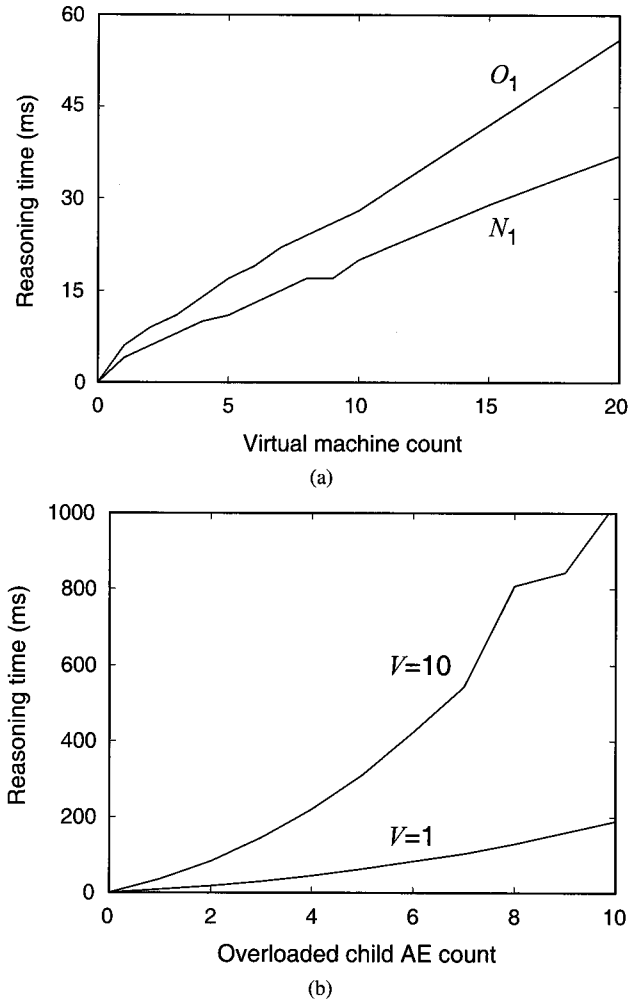

Fig. 7. The reasoning time for semantically matching filter rules with context: (a) Server AE level and (b) lowest data center AE level.

overloaded server within the cluster, the parent AE adds additional filter rules. The figure shows that when 10 servers within the cluster are overloaded, which each host 10 virtual machines, it takes up to 1 s to match all produced context with the generated filter rules. However, the figure also shows that when these servers only host a single virtual machine, and thus a lot less context is produced, the reasoning time is reduced to 200 ms. This means that reasoning time is much more heavily influenced by the amount of produced context than by the number of filter rules.

In summary, the presented intelligent context filtering approach has been shown to be capable of significantly reducing the amount of disseminated context during normal operations. Therefore, the reasoning time of the proposed semantic context dissemination processes can be kept in the order of tens of milliseconds. As problems occur within the network, more context needs to be distributed throughout the hierarchy, causing an increase in reasoning time. The presented results depict the context dissemination process in an AE hierarchy responsible for managing a 100,000 server data center. As long as the number of overloaded servers, and thus the amount of additional generated context information, remains somewhat low (i.e., below 50 per cluster), the reasoning time for generating new filter rules becomes at most a few seconds. If an entire cluster becomes overloaded, the reasoning time increased significantly up to 25 seconds. However, under these circumstances it is no longer necessary to generate additional filter rules, as all possible context is already being requested. The filtering process itself is more efficient, and even when up to 250 servers are overloaded, applying the generated filters takes less than 1 second. The results show that it is feasible to execute the filter rule generation process often (i.e., several times per second) during normal operations, and less often (i.e., once every few seconds) when a limited overload occurs, as the filter rule generation process only starts slowing down once these additional filters are in place. Additionally, the presented results show that even for a data center with up to 100,000 servers, of which a large portion is overloaded, the context filtering process can easily be executed once every second throughout the AE hierarchy.

VII. CONCLUSION AND FUTURE WORK

This article presents a hierarchical framework for efficient and scalable context dissemination in large-scale computing and communications systems. It focuses on intelligent and efficient collaboration and communication between distributed management components, referred to as AEs. Semantic models and reasoners are introduced to facilitate correct understanding and interpretation of information, goals, and actions. More specifically, we have shown how semantics can be employed to efficiently filter context, based on its meaning and the management goals of the AEs. As the managed environment changes, context filters are automatically and dynamically adapted by the presented framework. This dynamic process makes sure that the right context, is delivered to the right place, at the right time. Consequently, context-related overhead is significantly reduced, as superfluous information is not distributed among AEs.

The presented approach was applied to the management of large-scale cloud computing data centers. Managing such large-scale data centers presents a significant challenge, as huge amounts of context are generated by the underlying resources. The presented intelligent context filtering framework is therefore well suited to improve management scalability and efficiency in such a scenario. Additionally, the specific context requirements throughout the management hierarchy of this scenario were identified and it was shown how the framework can be applied to meet them. Finally, an analytical model was introduced to demonstrate how the proposed combination of AE hierarchies and intelligent context dissemination and filtering can be used to significantly reduce context overhead and support significantly increased scaling in terms of data center size.

A prototype implementation of the presented context dissemination framework was evaluated in order to estimate the effects of semantic reasoning on performance. The reasoning time of the filter rule generation and context filtering steps was measured in order to determine the limits of the context dissemination process. It was shown that under normal circumstances, both semantic generation and filtering take only a few tens of milliseconds in a data center of up to 100,000 servers. As more and more servers become overloaded, the filter rule generation process causes more detailed context to be disseminated

throughout the hierarchy. Consequently, the reasoning time increases. However, even in a highly overloaded network, where detailed context about up to 250 servers is distributed, filter rule generation takes only up to 25 seconds and actual filtering less than one second. It was thus shown that detailed context updates can be disseminated at one second intervals. Filter rule generation can be performed several times per second during normal operations (i.e., when it is most useful), and several times per minute once every few seconds during an overloaded period.

This article described the fundamental concepts of an intelligent context dissemination and filtering framework. In a fully autonomic context, the framework needs to automatically derive the dynamic context requirements at every level within the management hierarchy. These requirements depend on the specific management processes and algorithms used throughout the hierarchy. In future work, we are planning to close the gap between these management processes on one hand, and the context dissemination framework on the other, thus creating an integrated and fully autonomic solution.

REFERENCES

[1] B. Jennings, R. Brennan, W. Donnelly, S. Foley, D. Lewis, D. O'Sullivan, J. Strassner, and S. van der Meer, "Challenges for federated, autonomic network management in the future internet," in *Proc. IFIP/IEEE ManFI*, 2009, pp. 87–92.

[2] L. Baresi, A. D. Ferdinando, A. Manzalini, and F. Zambonelli, "The CASCADAS framework for autonomic communications," in *Proc. Autonomic Commun.*, 2009, pp. 147–168.

[3] J. Strassner, J.-K. Hong, and S. van der Meer, "The design of an autonomic element for managing emerging networks and services," in *Proc. ICUMT*, 2009, pp. 1–8.

[4] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in *Proc. CLOUD*, 2010, pp. 337–345.

[5] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud–protocols and formats for cloud computing interoperability," in *Proc. ICIW*, 2009, pp. 328–336.

[6] J. Strassner, J. de Souza, D. Raymer, S. Samudrala, S. Davy, and K. Barrett, "The design of a novel context-aware policy model to support machine-based learning and reasoning," *Cluster Comput.*, vol. 12, no. 1, pp. 17–43, 2009.

[7] J. Famaey, S. Latré, J. Strassner, and F. De Turck, "A hierarchical approach to autonomic network management," in *Proc. IFIP/IEEE ManFI*, 2010, pp. 225–232.

[8] W. K. Chai, A. Galis, M. Charalambides, and G. Pavlou, "Federation of future internet networks," in *Proc. IFIP/IEEE ManFI*, 2010, pp. 209–216.

[9] M. Serrano, S. van Der Meer, V. Holum, J. Murphy, and J. Strassner, "Federation, a matter of autonomic management in the Future Internet," in *Proc. IEEE NOMS*, 2010, pp. 845–849.

[10] K. Feeney, R. Brennan, J. Keeney, H. Thomas, D. Lewis, A. Boran, and D. O'Sullivan, "Enabling decentralised management through federation," *Comput. Netw.*, vol. 54, no. 16, pp. 2825–2839, 2010.

[11] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. B.-Yehuda, W. Emmerich, and F. Galan, "The RESERVOIR model and architecture for open federated cloud computing," *IBM J. Research and Development*, vol. 53, no. 4, pp. 1–11, 2009.

[12] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Proc. ICA3PP*, 2010, pp. 13–31.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[14] J. Keeney, D. Lewis, and D. O'Sullivan, "Ontological semantics for distributing contextual knowledge in highly distributed autonomic systems," *J. Netw. Syst. Manag.*, vol. 15, no. 1, pp. 75–86, 2007.

[15] J. Keeney, K. Carey, D. Lewis, and V. Wade, "Ontology-based semantics for composable autonomic elements," in *Proc. IJCAI*, 2005.

[16] J. Famaey, S. Latré, J. Strassner, and F. De Turck, "Semantic context dissemination and service matchmaking in future network management," *J. Netw. Manag.*, 2011.

[17] J. Wang, B. Jin, J. Li, and D. Shao, "A semantic-aware publish/subscribe system with RDF patterns," in *Proc. COMPSAC*, 2004, pp. 141–146.

[18] M. Petrovic, H. Liu, and H.-A. Jacobsen, "G-ToPSS: Fast filtering of graph-based metadata," in *Proc. WWW*, 2005, pp. 539–547.

[19] J. Skovronski and K. Chiu, "An ontology-based publish-subscribe framework," in *Proc. IIWAS*, 2006.

[20] H. Li and G. Jiang, "Semantic message oriented middleware for publish/subscribe networks," in *Proc. SPIE*, vol. 5403, 2004, pp. 124–133.

[21] M. Klusch, B. Fries, and K. Sycara, "OWLS-MX: A hybrid semantic web service matchmaker," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, no. 2, pp. 121–133, 2009.

[22] G. Shen, Z. Huang, Y. Zhang, X. Zhu, and J. Yang, "A semantic model for matchmaking of web services based on description logics," *Fundam. Informaticae*, vol. 96, no. 1, pp. 211–226, 2009.

[23] M. L. Sbodio, D. Martin, and C. Moulin, "Discovering semantic web services using SPARQL and intelligent agents," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 4, pp. 310–328, 2010.

[24] A. B. Bener, V. Ozadali, and E. S. Ilhan, "Semantic matchmaker with precondition and effect matching using SWRL," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9371–9377, 2009.

[25] H. Moens, J. Famaey, S. Latré, B. Dhoedt, and F. De Turck, "Design and evaluation of a hierarchical application placement algorithm in large scale clouds," in *Proc. IM*, 2011.

[26] J. Strassner, *Policy-Based Network Management—Solutions for the Next Generation*. Morgan Kaufman, 2004.

[27] L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu, "Towards a complete OWL ontology benchmark," *The Semantic Web: Research and Applications*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 4011, pp. 125–139, 2006.

[28] D. Harrington, R. Presuhn, and B. Wijnen. (Dec. 2002). An architecture for describing simple network management protocol (SNMP) management frameworks. RFC 3411 (Standard). Internet Engineering Task Force. [Online]. Available: http://www.ietf.org/rfc/rfc3411.txt

[29] J. Famaey, S. Latré, J. Strassner, and F. De Turck, "An ontology-driven semantic bus for autonomic communication elements," in *Proc. MACE*, Berlin, 2010, pp. 37–50.

[30] S. Latre, S. van der Meer, F. De Turck, J. Strassner, and J.-K. Hong, "Ontological generation of filter rules for context exchange in autonomic multimedia networks," in *Proc. IEEE/IFIP NOMS*, 2010, pp. 575–582.

[31] S. Davy, B. Jennings, and J. Strassner, "The policy continuum-policy authoring and conflict analysis," *Comput. Commun.*, vol. 31, pp. 2981–2995, Aug. 2008.

[32] M. Charalambides, P. Flegkas, G. Pavlou, and J. Rubio-loyola, "Dynamic policy analysis and conflict resolution for diffserv quality of service management," in *Proc. IEEE/IFIP NOMS*, 2006, pp. 294–304.

[33] F. Wuhib, R. Stadler, and M. Spreitzer, "Gossip-based resource management for cloud environments," in *Proc. CNSM*, 2010, pp. 1–8.

[34] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[35] M. J. Serrano, J. Serrat, J. Strassner, and M. Ó Foghlú, "Management and context integration based on ontologies, behind the interoperability in autonomic communications," in *Proc. CODS*, 2007.

[36] J. Strassner, J. N. Souza, S. van der Meer, S. Davy, K. Barrett, D. Raymer, and S. Samudrala, "The design of a new policy model to support ontology-driven reasoning for autonomic networking," *J. Netw. Syst. Manag.*, vol. 17, no. 1, pp. 5–32, 2009.

Jeroen Famaey obtained a M.S. degree in Computer Science from Ghent University, Belgium, in June 2007. Since August 2007, he is affiliated as a Ph.D. Student with the Department of Information Technology at Ghent University, where he is supported by a Ph.D. grant of the Flemish Institute for the Promotion and Innovation by Science and Technology (IWT-Vlaanderen). His main research interests include autonomic network management, semantic reasoning, and multimedia content delivery in broadband access networks. He was also involved in the European FP7 ALPHA project and the EUREKA CELTIC RUBENS project. Currently, he is participating in the European FP7 STREP OCEAN project.

Steven Latré obtained a Ph.D. and M.S. degree in Computer Science from Ghent University, Belgium, in 2011 and 2006, respectively. Since 2011 he is affiliated as a Post Doctoral Researcher with the Department of Information Technology at Ghent University, where he is supported by a grant of the Fund for Scientific Research Flanders (FWO-Vlaanderen). His main research interests include the use of autonomic communications to optimize the quality of experience management of multimedia services in broadband access networks. He was also involved in the IST FP6 project MUSE and EUREKA CELTIC project RUBENS and is currently participating in the FP7 STREP ECODE project as well as several other, national, projects.

John Strassner has over 35 years of experience. He is currently the Chief Technical Officer of the Software R&D Laboratory of the US Division of Huawei, where he leads autonomic system projects for managing cloud, enterprise, and service provider environments. He has served as a Professor of Computer Science and Engineering at the Pohang Univ. of Science and Technology and as a Visiting Professor at Waterford Institute of Technology in Ireland. Before that, he was a Motorola Fellow and Vice President of Autonomic Research at Motorola Labs. Previously, he was the Chief Strategy Officer for Intelliden and a former Cisco Fellow. He is a Distinguished Fellow of the TeleManagement Forum, and is currently the Chairman of the Autonomic Communications Forum. He is the past chair of the TMF's NGOSS SID, metamodel and policy working groups, along with the past chair of several IETF and WWRF groups. He has authored two books, written chapters for 5 other books, and has been co-editor of 5 journals dedicated to network and service management and autonomics. He is the recipient of the IEEE/IFIP Daniel A. Stokesbury memorial award for excellence in network management and the Albert Einstein for innovation in autonomic networking, and has authored over 275 refereed journal papers and publications.

Filip De Turck leads the network and service management research group at the Department of Information Technology of Ghent University, Belgium and the Interdisciplinary Institute of Broadband Technology, Flanders (IBBT). He received his Ph.D. degree from Ghent University in 2002 and his M.Sc. in Electronic Engineering from Ghent University in 1997. He was a Part-Time Professor from October 2004 till October 2006 and a Full-Time Professor since October 2006 in the area of telecommunication and software engineering. He is author or co-author of more than 300 refereed papers published in international journals and conferences. His main research interests include scalable software architectures for telecommunication network and service management, performance evaluation and design of new telecommunication architectures and services. In this research area, he is involved in several research projects with industry and academia, both on a national and European scale (FP7 projects). He is a Member of the Network Management Research Community by serving to Technical Program Committees of many network management conferences such as NOMS, IM, CNSM, APNOMS, and several workshops held in co-location.