INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2013; **00**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nem

A Semantic Context Exchange Process for the Federated Management of the Future Internet

Steven Latré[1]*, Jeroen Famaey[2] and Filip De Turck[2]

[1]*University of Antwerp - iMinds, Middelheimlaan 1, 2020 Antwerp, Belgium*
[2]*Ghent University - iMinds - Department of Information Technology, Gaston Crommenlaan 8/201, 9050 Gent, Belgium*

SUMMARY

In the Future Internet, federations are set up to cope with the stringent quality requirements of services. While a federated solution offers advantages in terms of scalability, it complicates the exchange of context (e.g., Quality of Service information of services) between federated nodes, as each node requires context to perform management tasks. In this article, we propose a context exchange process that automates the context communication between nodes. A scalable approach is proposed that is able to quickly react to local context updates, while maintaining a high level of expressivity to define relationships between federation partners. We distinguish between the context exchange inside an administrative domain, which focuses on scalability, and the context exchange between federation partners, which emphasizes the trust relationships between partners. In both cases, the process allows defining which context needs to be exchanged when and from where. Inside an administrative domain, a combination of RDF and SPARQL rules are used. This allows modeling the contextual requirements of management algorithms and automatically requesting remote context, only when it is necessary for the management algorithms to proceed. Between domains, an OWL-based approach is used, which allows describing the complex relationships between federation partners. Triggered by the intra-domain context exchange process, the contextual capabilities are communicated and refined through policies. Both type of processes are evaluated. The results show that they can infer which context is needed in a timely and scalable manner. As such, it outperforms approaches where context is broadcasted both in required bandwidth and end-to-end delay.

Received . . .

KEY WORDS: 5. Management Approaches / 5.2 Distributed management
6. Technologies / 6.6 Data, information and semantic modelling

1. INTRODUCTION

In recent years, the Internet has evolved from a best-effort packet forwarding service towards a service oriented delivery framework with often strict requirements in terms of Quality of Service (QoS) and Quality of Experience (QoE). This evolution results in important new challenges for the management of the Future Internet. The traditional, manual or assisted, network management performed today is not flexible enough to cope with the growing size, heterogeneity and complexity of the Internet in terms of services, devices and technologies. Therefore, the need has arisen for a more automated type of management, where autonomic managers are able to manage the networks' resources with no or little human assistance through the use of autonomic control loops.

*Correspondence to: University of Antwerp - iMinds, Middelheimlaan 1, 2020 Antwerp, Belgium. E-mail: Steven.Latre@intec.ugent.be

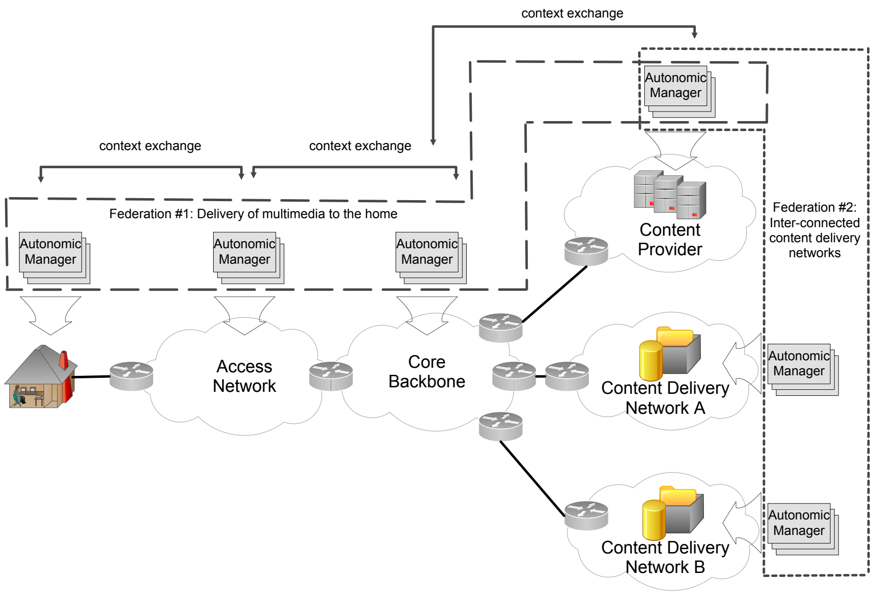


Figure 1. Overview of a federated architecture in a multimedia delivery environment. Two separate federations exist between different parties.

From a reasoning process point of view, a centralized management entity is optimal as this avoids instability in the decision process of the control loop, or worse, contradictory decisions. Although centralized management may be possible inside one administrative domain, the lack of a central authority in the Internet makes such management unfeasible in the overall Internet for both scalability and privacy reasons. Instead, several autonomic managers need to cooperate with each other in order to ensure end-to-end quality and cost optimized delivery of services. As illustrated in Figure 1, this cooperation is typically loosely coupled (e.g., the layered federation model of Feeney *et al.* [1]). Autonomic managers from different administrative domains form a federation in which they work together for a specific service delivery scenario. Note that it is also possible to have strict federations, where hierarchical relationships are introduced (e.g., the hierarchical model of Famaey *et al.* [2]).

To achieve an end-to-end management in cooperation (both in strict and loosely coupled federations), sharing of information is necessary to tackle and identify widespread problems (i.e., across the entire network) or to ensure that global optimizations can be enforced. We refer to context as all information that is relevant to the reasoning process of a control loop. Hence, to ensure an end-to-end delivery framework, there is a need for exchanging context between parties in a federation.

The exchange of context in a federation is challenged by two main factors. First, there is an issue of scalability. The contextual requirements of all autonomic managers can be large and diverse. Moreover, different autonomic managers can have different contextual requirements and these requirements can change over time. It is thus important to accurately define which context needs to be exchanged from where and when. Second, between administrative domains the context exchange is further complicated because of trust and privacy concerns. There is thus a need for a process that decides which context can be exchanged to which parties. Current context exchange processes typically rely on static solutions, where the type of information that is exchanged is fixed. For example, the paradigm of content-based networking (CBN) [3] defines the exchange of context through subscriptions in the publish-subscribe paradigm. However, the subscription filters are static and cannot change without any manual intervention. Furthermore, they also feature only a syntactic definition of context: it is not possible to infer which context is best suited where. Initial semantic approaches have been proposed (e.g., Keeney et al. [4]) but further work is needed to come to an automated and semantic context exchange process.

In this article, we propose a framework that tackles the above challenges by introducing a process that automates the exchange of context. The contributions of this article are the following. First, we allow to automatically derive which context needs to be requested from where. We use an approach where the contextual requirements of each federation party are modeled in RDF[†]. This allows a fast and timely reasoning based on SPARQL[‡] and SPIN[§], which can scale to large-scale problem domains. Second, we introduce the use of a semantic model to improve the context exchange between parties in the federation. Based on OWL-2[¶] and SWRL[||] inferencing, this model is able to refine the standard context exchange between parties based on the modeled relationships in the federation. Third, we propose an inter-domain context exchange process that decides on which context can be made available to other parties in the federation. The process is linked with the existing WS-Agreement [5] specification, which can define agreements between parties in the federation.

The remainder of this article is structured as follows: Section 2 describes the federated architecture we use in our work. An illustrative use case of multimedia delivery that will be used throughout the article is discussed in Section 3: we detail which context needs to be exchanged in this use case and discuss the main challenges. In Section 4, we discuss how our approach combines both domain-specific and domain-neutral information models to represent context. Section 5 and 6 discuss the context exchange process itself, focusing on intra and inter domain context exchange, respectively. Section 7 presents detailed evaluation results of the performance of the proposed approach. Related work in the field of context modeling in network management and the use of federations in the Future Internet is discussed in Section 8. Finally, Section 9 concludes this paper.

2. EXCHANGING CONTEXT IN A FEDERATED ENVIRONMENT

In this section, we discuss the role of exchanging context in a federated environment but first define the terms federation and context. We define the term federation according to Feeney *et al.* [1]: *"A federation is considered to be a persistent organizational agreement, which enables multiple autonomous entities to share capabilities in a controlled way."* A federation thus groups different organizations, consisting of multiple autonomous entities, to cooperate with each other by sharing capabilities (e.g., resources, information) with the goal of an improved management. To share capabilities, the autonomous entities need to exchange context. We use the following definition of context from DEN-ng [6]: *"The context of an entity is a collection of measured and inferred knowledge that describe the state and environment in which an entity exists or has existed."* In particular, this definition emphasizes two types of knowledge: facts (that can be measured) and inferred data, which results from machine learning and reasoning processes applied to past and current context.

The typical setup of a federation is illustrated in Figure 2. As shown, different sequential steps are taken to construct this federation. We discuss them in more detail below in this section and focus on the implications for the context exchange process.

2.1. Service matchmaking

Figure 2 shows a scenario where one partner, the federation initiator, wants to consume a service offered by another partner. In the scenario illustrated in Figure 2, similar management services are offered by different providers (Service Provider A and B) and the federation initiator first needs to find which federation partner, matches best with the requested service. This process is called service matchmaking. In previous work, we have proposed a semantic matchmaking algorithm [7], where

[†] Resource Description Framework (RDF) - http://www.w3.org/RDF/
[‡] SPARQL Query Language for RDF - http://www.w3.org/TR/rdf-sparql-query/
[§] SPIN - SPARQL Syntax: http://www.w3.org/Submission/2011/SUBM-spin-sparql-20110222/
[¶] OWL 2 Web Ontology Language Document Overview: http://www.w3.org/TR/owl2-overview/
[||] SWRL: A Semantic Web Rule Language Combining OWL and RuleML - http://www.w3.org/Submission/SWRL/

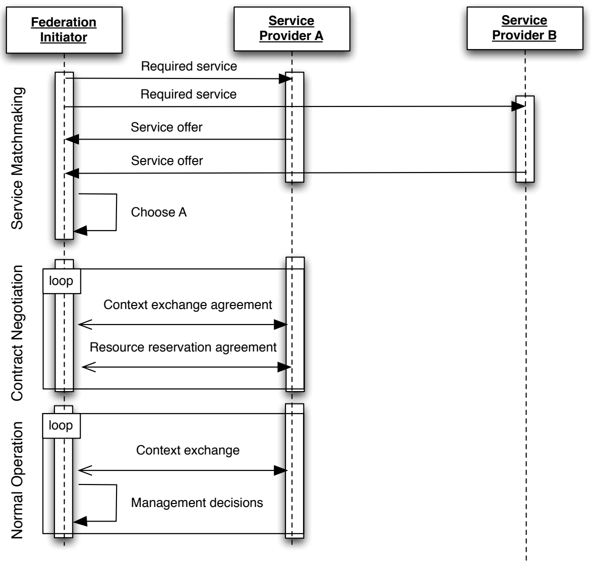


Figure 2. Setup of a federation across different administrative domains consisting of a service matchmaking process, a contract negotiation process and the normal operation, after the federation has been set up.

the offered and requested services are described in OWL. This allows reasoning on the semantics of the offered and requested services as opposed to a pure syntactic matchmaking algorithm.

2.2. *Contract negotiation*

Once the ideal federation partner has been found, a formal contract needs to be defined between the two partners. Such a contract defines the rights and duties of both parties in the federation. It describes which services can be accessed and what the service level objectives are. Protocols exist to perform this contract negotiation (e.g., WS-Agreement [5]).

The contract negotiation process consists of two types of negotiation, both of them can be specified using WS-Agreement. On one hand, the parties need to agree on the context they will exchange. This agreement process is dependent on the type of services offered and requested, the internal contextual requirements for this service and will be influenced by the policies set by the context provider. In this article, we present such a context agreement process in Section 6, which is integrated with the WS-Agreement specification.

2.3. *Normal operation*

Once a formal contract has been agreed between the federation partners, the federation has been set up and the normal federation operation can begin. In terms of management tasks, the exchange of context is the most important communication between the federation partners. Based on this context, each federation partner is able to carry out its own management decisions. As mentioned, we can distinguish between two types of context exchange: context exchange that falls inside the same administrative domain and context exchange that does not. There are notable differences between these two and we discuss both of them in more detail.

2.3.1. Context exchange inside an administrative domain In this case, the different autonomous entities fall under the same scope of authority, which results in a high level of trust between the federation partners. Consequently, the setup of the federation and the exchange of context inside an administrative domain can be simplified. The same information model will be used between autonomous entities. As opposed to the context exchange between administrative domains, low-level context information will be required and the autonomous entities will need to respond quickly to anomalies such as network and device failures. For example, if packet loss occurs in the access

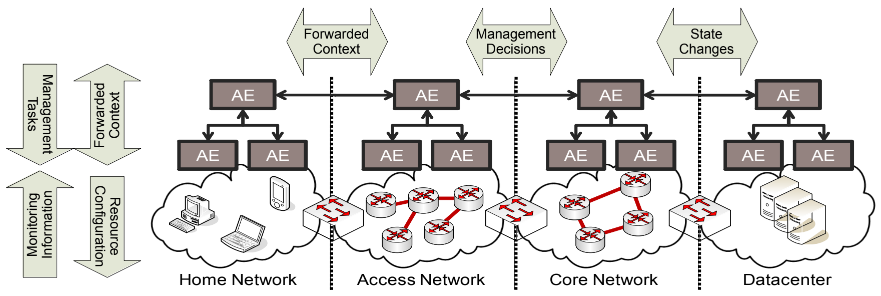


Figure 3. Overview of a network management environment, after the setup of a federation. Arrows represent context exchange between the Autonomous Entities (AEs).

network, the autonomic management framework will need to respond quickly to the potential QoE drop by detecting, locating and solving the problem as fast as possible.

The main challenge in this case is thus to cope with the large amount of context and to organize the context exchange in a scalable and timely way. Flooding the administrative domain with all the available contextual data is sub-optimal and in some large scale problems infeasible. We present an algorithm that automatically deduces the required context in Section 5 based on an RDF dataset and SPARQL rules.

2.3.2. Context exchange between administrative domains Between administrative domains, context exchange is complicated by the lack of a central authority and only a limited level of trust between the federation partners. The main challenge in this case is therefore coping with this lower trust level. In Section 6, we describe an OWL-based model and reasoning process that allows dealing with these restrictions.

3. USE CASE: MULTIMEDIA VIDEO DELIVERY

To provide an illustrative example of how the context exchange process is used in a federation, we describe the exchange of context for an important use case, which will be used throughout the remainder of this article. In this use case we assume that a set of critical cloud applications, offering multiple multimedia services to its end-users, are hosted on a remote cloud server. In order to ensure a quality optimized multimedia delivery to the users, the video connections need to be managed end-to-end. To offer such an end-to-end network management, we assume that a federation has been set up as illustrated in Figure 3. In this use case, there are several administrative domains: the home network, the access network, and multiple core networks (represented in the figure as a single domain for the sake of simplicity) and the cloud provider. We assume that the cloud provider and access network have a preferred relationship. This means that the cloud provider is willing to share more to the access network than to the other partners in the federation and vice versa. We focus on the context exchange process of the cloud infrastructure's root autonomous entity. This autonomous entity communicates with the root autonomous entities of the other administrative domains and its child autonomous entity.

As the primal provider of the multimedia service, the cloud provider has a set of management tasks at hand that allow optimizing the service delivery (e.g., load balancing, resource reservation). In order to perform these tasks, the cloud provider requires context from the other federation partners. For example, it requires a video quality score from the home network that describes how the video is perceived by the end user.

Figure 4 details the context exchange process in the normal operation step. We assume that the context exchange process has initially been configured as follows: the cloud provider's control loop initially only requires basic monitoring information. This context is used to trigger the detection of a lack of resources in the network or overloaded servers in the cloud. The context requests specify

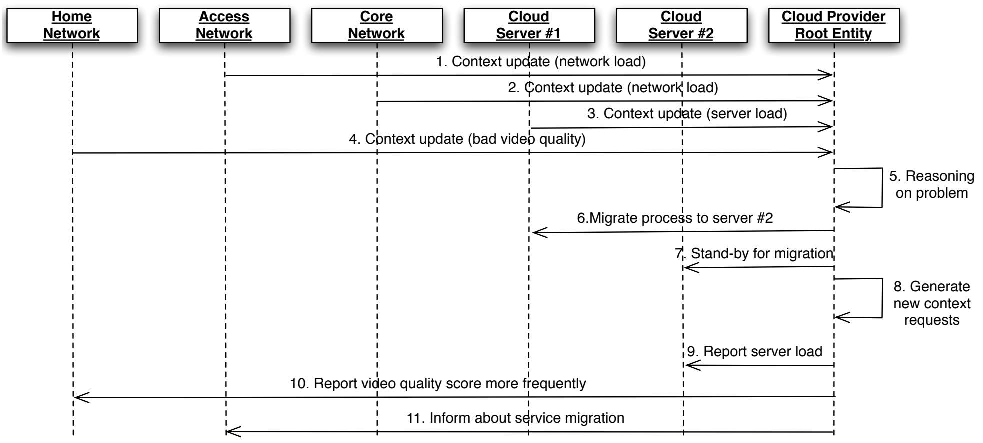

Home Network
Access Network
Core Network
Cloud Server #1
Cloud Server #2
Cloud Provider Root Entity
1. Context update (network load)
2. Context update (network load)
3. Context update (server load)
4. Context update (bad video quality)
5. Reasoning on problem
6.Migrate process to server #2
7. Stand-by for migration
8. Generate new context requests
9. Report server load
10. Report video quality score more frequently
11. Inform about service migration

Figure 4. Sequence diagram illustrating the exchange of context in a federation with a cloud provider, core network, access network and home network. The sequence diagram illustrates how a context change triggers new context requests.

that the monitored video quality score on the home network and the network and server load need to be forwarded to the cloud provider on regular intervals (e.g., every second).

There are many circumstances in which these initial context requests may change: one of them is a change in context triggered by either a locally or remotely monitored value. For example, suppose that a cloud server becomes overloaded. Figure 4 illustrates the sequence of events that occur, starting from the initial change in context up to the deployment of a new set of context requests. As stated, the cloud infrastructure's autonomous entity receives regular, aggregated, context updates of each of the partners (steps 1-3). We assume that, at a given time, the home network detects a bad video quality offered from one of the multimedia applications in the cloud (step 4), caused by the overloaded server. This bad video quality score triggers a reactive reasoning process in one of the control loops of the cloud provider that tries to solve this problem (step 5). In this case, the problem is solved by migrating one of the applications to a second cloud server. The process migration is initiated from server #1 (step 6) to server #2 (step 7). Through reasoning in the context exchange process, new context requests are generated based on the change in the management decisions (step 8). In this case, two new context requests are generated. First, the cloud provider informs the lower-level cloud servers about all changes and also asks the new server to start reporting its load as well, as it wants to monitor its performance (step 9). Second, because a problem has occurred, the control loop of the cloud provider wants to closely monitor the delivered quality and instruct the home network to send updates about the video quality score at a higher frequency (step 10). Finally, the cloud provider updates the access network about the performed management decisions (step 11). Although this information might be useful to other partners as well, the cloud provider only forwards this data to the access network as it has a preferred relationship with it and thus a higher level of trust.

4. OVERVIEW OF USED INFORMATION MODELS

To describe the contextual requirements, a combination of both domain-specific and domain independent models is used. Figure 5 illustrates how these models are organized and for which scenario they are used.

Furthermore, because of the lack of a central authority, different partners in a federation will typically also utilize different models. There should be a semantic translation possible between these models. Note that this ontological mapping between different inter-domain models is out of the scope of this paper as several approaches have been proposed in the past [8, 9]. In the context of ontology mapping, there are three main approaches that can be observed,. The Global as View

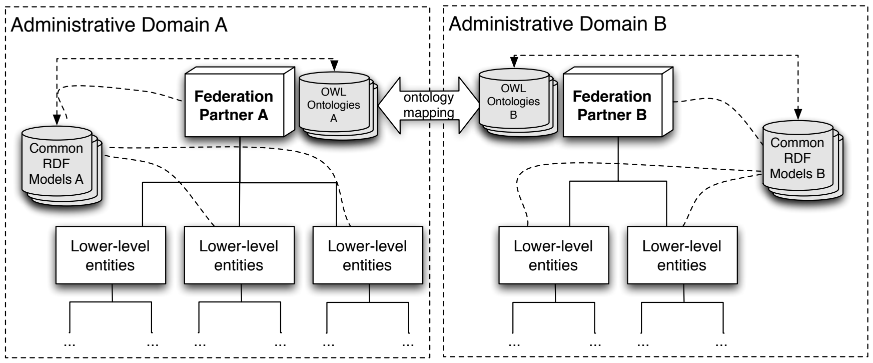


Figure 5. The organization of information models in our federated context exchange process. Entities inside one domain share a common RDF dataset. An OWL model is used for interacting with federation partners across domains. An ontology mapping algorithm is typically required to map between the ontologies.

(GAV) approach [10] expresses the global database as a function of local databases. The Local as View (LAV) approach [11] on the other hand, describes each local schema as a function over the global schema. This has as advantage that new sources can be added independently of the other sources. The Global and Local as View (GLAV) [12] approaches combine both principles and define specific peer-based mappings to link two ontologies. As a federation typically also relies on peer-based cooperation, the GLAV approach is most promising for our solution, as it also features the highest extendibility.

4.1. *Intra-domain models*

Inside an administrative domain, a set of RDF datasets is used. In our approach, these datasets must describe the following aspects: (i) an overview of the context that is available (both the context types and the contextual data itself), (ii) an overview of the resources that are managed by the specific autonomous entity and (iii) information about the reasoning process of the control loops of the management nodes and their contextual requirements available in the different autonomous entities.

4.2. *Inter-domain models*

Between administrative domains, a higher level of expressiveness is required. Therefore, a set of OWL ontologies is used to model the context exchange in this case. In our approach the OWL models must at least describe the following aspects: (i) the relationships between the federation partners that are relevant for the context exchange process and (ii) the context types that can be exchanged between partners in the federation and (iii) the contextual requirements relating to context available outside the current domain and derived by the context exchange process inside the administrative domain.

4.3. *Models used for representing context*

In the remainder of this section, we discuss the models that were used to represent both the RDF and OWL models. However, it should be noted that these models are examples related to the use case described in Section 3 and other models can be used as long as they adhere to the requirements stated above.

4.3.1. Intra-domain RDF models In our approach, the common RDF model consists of both domain-specific sub-models and domain-neutral sub-models. To represent the managed environment, we use a subset of *DEN-ng* [6] that focuses on the context-related aspects of the environment. DEN-ng is used to represent the physical and logical state of the network and its resources, as well as the business goals and internal workings of the governing organizations.

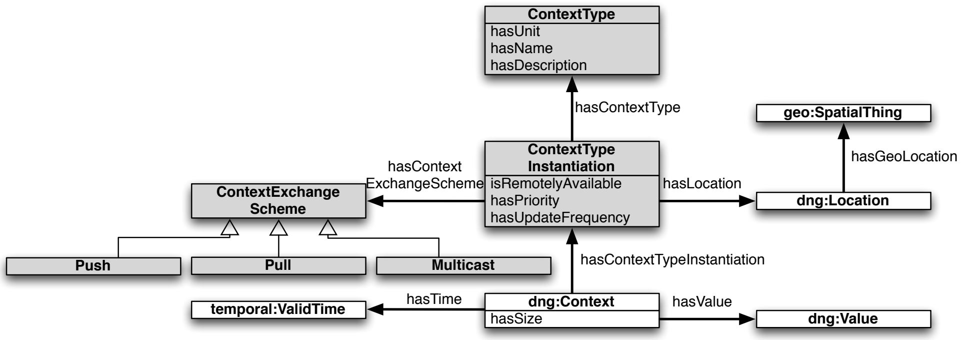


Figure 6. Illustrative model used for representing contextual data, the context types and their instantiation. The integration with existing models is illustrated. The grey boxes highlight the proposed extensions.

To model the context types and contextual data available, the DEN-ng model is extended with a more elaborate *context model*. This novel context model and its integration with the other models is illustrated in Figure 6. As shown, the model distinguishes between three levels of describing context. The DEN-ng model already has a notion of contextual data (i.e. the `Context` class), which describes actual values of a certain context type. In our model, we link this contextual data with a `ContextTypeInstantiation`, which is in turn linked with a `ContextType`. A `ContextType` models the category of contextual data we are referring to. One `ContextType` can have multiple `ContextTypeInstantiations` that link a `ContextType` to an actual location. For example, suppose we have a context type 'video quality' that describes the observed video quality on a client device. There will be only one video quality `ContextType` but this quality can be observed on multiple client devices. Therefore, there will be one `ContextTypeInstantiation` per client, which all link with a single `ContextType`. Over time, the video quality can be measured continuously. There will be multiple instances of the stored contextual data that link with a single `ContextTypeInstantiation`. Furthermore, a `ContextTypeInstantiation` also adds some additional semantics to a `ContextType`, which is specific to the autonomous entity where it is described. These semantics entail its priority, the preferred exchange schemes, its update frequency and whether or not it can be made available to other entities outside

A final part of the used RDF models is the description of the reasoning process of the control loops. In our approach, we use the *Rule Interchange Format Production Rule Dialect (RIF-PRD)* vocabulary [**]. RIF-PRD is a format to describe production rules (i.e., IF-THEN rules) in a standard way that allows interoperability between parties. We use RIF-PRD to describe the inputs and outputs required by a control loop.

4.3.2. Inter-domain OWL models The inter-domain OWL model uses the same *context model* as was described in Section 4.3.1. Note that, the data stored in the inter-domain model will be a subset of that in the intra-domain model. How this subset is determined is discussed in Section 6. Second, to model the contracts between partners in a federation, the *WS-Agreement* specification was modeled in OWL. This allows modeling concepts such as service level objectives and guarantees. In our approach, the WS-Agreement model will be used to automatically derive the exchange of context between parties into WS-Agreement requirements. Additional sub-models can be used, i.e. to model trust, relationships between organizations, etc.

[**]RIF Production Rule Dialect - http://www.w3.org/TR/rif-prd/

Table I. Compact SPARQL notation used throughout this article.

SPARQL QUERY	Simplified notation
CONSTRUCT A WHERE B	B → A
?ns1:a ns2:hasRel ?ns3:b	hasRel(a,b)
Statement1 Statement2 Statement3	Statement1 ∧ Statement2 ∧ Statement3

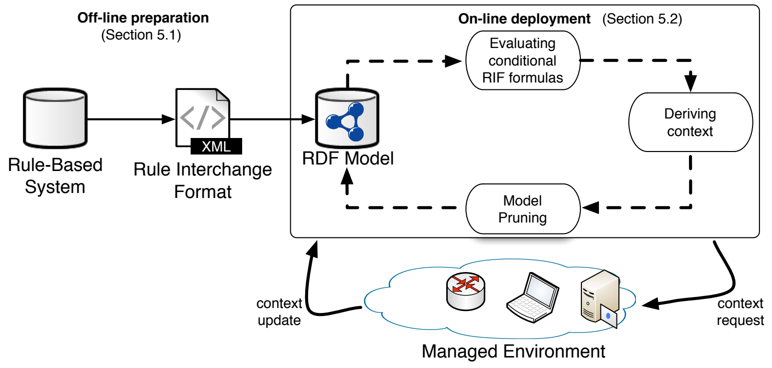


Figure 7. Overview of the context exchange process inside an administrative domain.

5. INTRA-DOMAIN CONTEXT EXCHANGE PROCESS DETAILS

The goal of the context exchange process inside an administrative domain is to decide which context needs to be requested when, with the goal of avoiding the flooding of the distributed federation parties. The operations that are supported are the following: if, at a given time, federation parties require a remote context item to continue their reasoning process, this context item must be requested.

Figure 7 provides an overview of the intra-domain context exchange process. As illustrated, it consists of two main phases. In an off-line phase, the information in the control loop (e.g., a rule-based system) is translated into the RDF model. In a second and on-line phase, the modeled information is continuously compared with the current contextual state of the system.

Both the intra- and inter-domain context exchange processes extensively use SPARQL perform the automated inference of which context is needed. Throughout this article we use a more compact notation of SPARQL rules and queries as illustrated in Table I. A number of simplifications are made to this notation without losing generality, to avoid too lengthy notations.

5.1. Off-line construction

In order to understand the contextual requirements of the control loop, the RDF model must contain information about the reasoning process. As shown in Figure 7, we translate a rule-based system into the interchangeable RIF format, which produces an XML file with all the rules, and then ultimately into the RDF model. Both translations occur completely automated.

After performing a pure syntactic translation from RIF to compliant RDF, the translation step links the rules with the actual context information (and more specifically to a `ContextTypeInstantiation`) described in the context model. Therefore, the translation looks for all variables in the RIF rules (belonging to the `Var` class) and links them to the corresponding contextual data.

Some of the linked context may be locally available, while other is only available from remote autonomous entities. The translation step determines whether one of the variables is locally or remotely available through the following SPARQL rule:

$$\begin{aligned} type(this, Var) \wedge hasContextTypeInstantiation(this, type) \\ \wedge hasLocation(type, loc) \wedge isOwnLocation(loc, true) \rightarrow isLocal(this, true) \end{aligned} \tag{1}$$

Here, the `type` property defines the classical RDF type property for class membership. This rule retrieves the location of the `ContextTypeInstantiation` instance that is linked with the variables of a RIF rule and identifies the variable as local (denoted by `isLocal`) if these instantiations correspond with their own location (denoted by `isOwnLocation`). A similar rule marks the variable as remote. Based upon the locality of a variable we can derive the locality of an atomic formula in a RIF rule. We define an atomic formula to be remote if at least one variable is remote. This is characterized by the following SPARQL rule:

$$hasVar(this, var) \wedge isRemote(var, true) \rightarrow isRemote(this, true) \tag{2}$$

The `hasVar` property is a property, which links a RIF formula with the variables they contain and is generated by the RIF translation.

5.2. On-line deployment

Based on the rule information and its link with the context model of the RDF model it is possible to derive which context needs to be requested when. To enable this, a three-step approach is taken that is triggered each time new contextual data is stored in the RDF model, which is discussed in the remainder of this section.

5.2.1. Step 1: Evaluating conditional RIF formulas When new context is added to the RDF model, the context exchange process is triggered to investigate the state of the conditional formulas in the RIF rules. This state corresponds with the truth value of the formulas. The truth value of a formula denotes whether the formula is true, given the current state of the environment. To be able to know which context needs to be requested, the context exchange process needs to know which rules from the federation partner can potentially be triggered. Therefore, the RDF model tries to derive the truth value of all formulas.

To derive this, we present SPARQL rules that formally define how the truth values of conditional formulas in a RIF rule must be interpreted. We start with deriving the truth values of a local atomic formula and then describe the possible combinations of a RIF rule.

Atomic formula The following SPARQL rule allows characterizing whether a local atomic formula evaluates to true or not, given the current contextual state of the RDF model.

$$\begin{aligned} type(this, Context) \wedge hasContextTypeInstantiation(this, ins) \wedge insHasVar(ins, var) \\ \wedge hasAtom(var, atom) \wedge isLocal(atom, true) \wedge hasVarList(atom, varList) \\ \wedge t = evaluateAtom(atom, varlist) \rightarrow evaluatesToTrue(this, t) \end{aligned} \tag{3}$$

The above rule introduces an additional property `evaluatesToTrue` that defines the truth value of the atomic formula. The `insHasVar` property links a context type instantiation with the corresponding RIF variable and is defined by the translation step in the previous section. Furthermore, the `hasAtom` is a RIF-based property and links variables with the atomic formulas that define them. The `evaluateAtom` function is a custom SPIN function that allows investigating the truth value of an atomic formula given a list of variables. This function takes as argument the atomic formula to evaluate (`atom`) and an optional list of variables (`varlist`). This list of variables is used to evaluate an atomic formula, which is part of an existential formula. For example, in the RIF formula $\exists x, y \quad someProperty(x, y) \wedge otherProperty(y)$, the two atomic formulas $someProperty$ and $otherProperty$ will be evaluated with the list of variables x and y. This list defines that the variables x and y do not directly link to context, but link to variables, which were defined by an existential operator.

Table II. SPARQL rules for deriving the truth value of RIF formulas.

RIF Formula	SPARQL Rule
Conjunction	$type(this, And) \wedge hasAtom(this, atom) \wedge NOTEXISTS(isRemote(atom, true))$ $\wedge NOTEXISTS(evaluatesToTrue(atom, false)) \rightarrow localEvaluatesToTrue(this, true)$
Negation	$type(this, Not) \wedge hasAtom(this, atom) \wedge NOTEXISTS(isRemote(atom, true))$ $NOTEXISTS(evaluatesToTrue(atom, true)) \rightarrow localEvaluatesToTrue(this, true)$
Existential	$type(this, Exists) \wedge formula(this, formula) \wedge localEvalutesToTrue(formula, true)$ $\rightarrow localEvalutesToTrue(this, true)$ $type(this, Exists) \wedge hasVarList(this, varList) \wedge formula(this, formula)$ $\rightarrow hasVarList(formula, varList)$

Combining formulas Once the truth value of a local atomic formula has been determined, it is possible to investigate the combinations thereof. However, we can only say something about the locally available context and not about the remotely available context. Therefore, we introduce an additional property, `localEvaluatesToTrue`, next to the `evaluatesToTrue` property defined above. For each combined formula, we can formally define the semantics of the `evaluatesToTrue` and `localEvaluatesToTrue` properties. The `localEvaluatesToTrue` property describes the truth value of the local formulas only. In general, we attach the `evaluatesToTrue` property only if all atomic formulas are local. In contrast, the `localEvaluatesToTrue` property needs to investigate for each atomic formula its locality and only take the local atomic formulas into account. Defining the `evaluatesToTrue` property can thus be seen as a special case of the `localEvaluatesToTrue` property in which all atomic formulas are local. For the sake of brevity, we therefore focus on the `localEvaluatesToTrue` property only.

To evaluate the truth value of a disjunction, we can use the following reasoning. A disjunction is true if at least one part of the disjunction is true. Hence, the following simple rule can derive the `localEvaluatestoTrue` property for a disjunction.

$$\begin{aligned} &type(this, Or) \wedge hasAtom(this, atom) \wedge NOTEXISTS(isRemote(atom, true)) \\ &\wedge evaluatesToTrue(atom, true) \rightarrow localEvaluatesToTrue(this, true) \end{aligned} \tag{4}$$

With a similar reasoning, the `localEvaluatesToTrue` property can be derived for the other type of RIF formulas, being conjunction, negation and existential formulas. A conjunction is true only if there are no components of that formula, which are false. A negation is true if there is no statement that says that the formula to be negated is true. Finally, an existential formula is true if its subformula is true, given the variables that it introduces (denoted by `varList`). The corresponding SPARQL rules are summarized in Table II.

5.2.2. Step 2: Deriving context Once the truth value of every atom in the RIF rules has been determined, we can derive whether it is necessary to request additional remote context. In general, context should be requested if (i) it is part of the condition of a rule and (ii) whether the rule is fired depends on the value of that remotely available context.

For example, suppose `a` and `b` contain references to local context type instantiations and `r` is a reference to a remote context type instantiation. For a rule consisting of a conjunction in the condition part (e.g, `IF a AND b AND r THEN c`), the context type instantiation `r` should only be requested when both `a` and `b` are already true as this is the only time this rule would be triggered. Similarly, if the condition of a rule consists of disjunctions, a remote context type instantiation needs to be requested if all local context types lead to false. This means that the truth value of that remote instantiation can lead to the triggering of that rule.

To enforce this desired behavior, we defined a series of SPARQL rules. First, for each type of formula, we define its impact on the context exchange process. For example, for a conjunction we can define the following SPARQL rule, that implements the desired behavior discussed above.

$$\begin{aligned}
&type(this, And) \wedge formula(this, formula) \wedge localEvaluatesToTrue(this, true)\\
&isRemote(formula, true) \wedge hasContextTypeInstantiation(formula, ins)\\
&\wedge hasContexttypeInstantiation(context, ins) \wedge hasTime(context, time)\\
&\wedge FILTER(isDeprecated(time, formula)) \rightarrow requestContext(ins, true)
\end{aligned} \quad (5)$$

The above rule introduces the `requestContext` property to identify the context type instantiations that need to be requested. Here, the `formula` property links a conjunction with the sub-components of this conjunction. Note that the above rule checks if a local context value exists, which is still relevant to the rules of the control loops. the `isDeprecated` is a dedicated function, which performs a simple numeric comparison on the time value (through the `hasTime`property). This numeric comparison checks whether the most recent context value that is stored in the RDF dataset still falls in the temporal scope defined by the variables in the formula. For the other formula types (disjunction, negation and existential), similar SPARQL rules can be constructed. Once the `requestContext` property has been correctly assigned, querying which context needs to be requested is straightforward:

```
SELECT ?c WHERE { ?c :requestContext true }
```

Note that the above approach introduces a multi-phase context acquisition process where first only a summary of the status of the managed network is requested and new context is requested if the summarized data signals abnormal behaviour (e.g., a sudden drop in QoS parameters). This corresponds to the common practice in today's management, which is often based on the signaling of alarms. If an alarm is received, additional management tasks are started such as fault localization [13] and root cause analysis [14]: both of these tasks typically require additional contextual data. The proposed approach can introduce a delay in some pathological cases if the context exchange process itself is suffering from network anomalies (e.g., congestion on the control channel). It is up to the network manager to decide whether this delay can be tolerated for a particular subset of RIF rules. There is thus a trade-off between always sending all context and selectively requesting context in multiple phases as discussed above. For high priority management tasks, which require the fastest (i.e., sub-second) reaction time, the first option will be most suitable. In general, the latter is suitable when (i) the amount of contextual data to request in subsequent phases is large and (ii) the probability of anomalies (triggering the request of new context) is low. We argue that the majority of management tasks will benefit from the proposed multi-phase approach as they adhere to the above conditions and typically do not require that strict reaction times (i.e., severe network anomalies are rare). Moreover, the proposed approach allows splitting the context exchange behaviour from the actual management tasks. Therefore, it allows a network manager to set the above described policies to differentiate between high-priority tasks and other and hence achieve the above discussed behavior.

The above described SPARQL rules in both steps can be used in two configurations. On the one hand, if the SPARQL rules are directly applied as stated above, if new contextual data is stored, the reasoner will investigate the status of all RIF rules to evaluate their truth value. Alternatively, it is possible to define a form of Event-Condition-Action (ECA) rules in SPARQL that allows evaluating only the subset of rules that are directly linked with the updated contextual data. In our approach, such an ECA structure is achieved through SPARQLMotion ††, which is an RDF-based scripting language mainly intended for data processing workflows. It defines several modules which can sequentially be executed. Through a special purpose module, new data that is added to the model can be identified. By using SPARQLMotion, each time contextual data is added, a module recursively finds the RIF rules that link to that context (through either atomic or combined formulas). The above SPARQL rules are then defined in two separate modules that alter the model as discussed, but only on the relevant subset of the model. We discuss the performance of both options in Section 7.

††SPARQLMotion - http://sparqlmotion.org/

5.2.3. Step 3: Pruning the dataset The context exchange process continuously investigates the context information stored in the RDF dataset to determine the context that can be requested. Therefore, newly available contextual data is continuously being stored and linked in the dataset. However, the RIF rules typically only refer to contextual data with a limited temporal scope. This means that, as more data is being stored in the dataset, some of the data can become redundant. To avoid that the RDF dataset becomes flooded with redundant historic data we continuously prune the dataset by removing this historic data. This is done by finding the contextual data that is not referred to by the variables.

5.3. Intra-domain context exchange in the use case

To illustrate how the above context exchange process works for a specific scenario, we provide an example of how context requirements in the multimedia video delivery use case are translated and inferred. Consider the following rule also discussed in Section 3: ”If the delivered quality of a video hosted from a server is bad and the CPU load or memory load on that server is high and another server is available with sufficient load, then the video should be migrated to that other server”. Or, more formally:

```
isBad(VideoQuality,Server1)
AND (cpuLoadIsHigh(Server1CPULoad)
OR memoryLoadIsHigh(Server1MemoryLoad))
AND cpuLoadIsLow(Server2CPULoad)
AND memoryLoadIsLow(Server2MemoryLoad)
```

During an offline phase, this rule is first translated into the RDF dataset as an RIF statement. In doing so, terms such as `Server1CPULoad` and `VideoQuality` are linked with the appropriate context type instantiations in the context dataset. Additionally, the locality of each of these variables is determined. As the video quality will typically be locally available (as it is already forwarded to the cloud provider), this is considered as local context, while the other variables are marked as being remote.

Suppose that at some given time, a new video quality score is stored into the RDF dataset containing a low value. This new event triggers the SPARQL rule defined in Equation 3 and will attach the `localEvaluatesToTrue` property to the local atomic formula `isBad(VideoQuality,Server1)`. This will again trigger a series of SPARQL rules (i.e., the conjunction and disjunction SPARQL rules defined in Table II and the SPARQL rule defined in Equation 5, which ultimately leads to attaching the `requestContext` property to the corresponding context items. Hence, as the video quality is bad, additionally context items are requested to assess delve deeper into the problem. However, if the `localEvaluatesToTrue` property was not true (e.g., because the video quality was high), no additional context would be requested as the rule would not fire, regardless of the value of the remote context.

6. INTER-DOMAIN CONTEXT EXCHANGE PROCESS DETAILS

The inter-domain context exchange process determines which context types can be exchanged between the federation partners and fixes this in a contract, defined through a WS-agreement document. In contrast with the intra-domain context exchange process, the focus is not on the scalability of the context exchange but on trust and privacy concerns. The inter-domain context exchange process is only executed during the construction of the contract between federation partners. It communicates the context exchange capabilities between these partners. The process consists of three distinct phases, which are discussed in the next section.

6.1. Step 1: Advertising contextual requirements

As discussed in Section 5, the intra-domain RDF dataset contains all context type instantiations that might be of interest to the control loops residing in the different autonomous entities inside the domain. In this step, the intra-domain RDF dataset of each autonomous entity is queried and the context type instantiations belonging to the final category are collected. This is done through the following SPARQL query:

$$\begin{aligned} type(context, ContextTypeInstantiation) \wedge isRemote(context, true) \\ \wedge OPTIONAL(partOfOwnManagementDomain(loc, false)) \\ \wedge OPTIONAL(hasLocation(context, loc)) \\ \wedge type(ownloc, Location) \wedge isOwnLocation(ownloc, true) \rightarrow context, loc, ownloc \end{aligned} \tag{6}$$

The above query selects all context type instantiations that are remote to the queried autonomous entity and of which the location does not belong to the administrative domain currently queried. Moreover, it also queries the location where it can request this context and it's own location as a reference. Note that this query also selects items with a missing `hasLocation` property. In this case, the autonomous entity does not know the exact location of where the context is available and it assumes a remote availability. The returned context type instantiations by the SPAQRL query can be used to request the exchange of context through the WS-Agreement Negotiation protocol. Once the required context type instantiations are determined, a WS-Negotiation offer is generated that requests the corresponding context type instantiations, with their attached guarantees. As discussed in Section 4, these guarantees are part of the context model. The generation of the WS-Agreement Negotiation offer itself is straightforward as it is a pure syntactic translation, which maps the queried context (i.e., the `context` variable in SPARQL rule 6) and its guarantees to the corresponding XML-based syntax.

6.2. *Step 2: Matching contextual requirements*

The generated offer will be sent to several autonomous entities. Each partner in the federation receives the advertisements of contextual requirements through WS-Agreement Negotiation offers from the other partners in the federation. If a partner has a requested context type available, its request for it is stored in the OWL model through the `requestedBy` property.

Only the entities that have the requested context can positively reply with a counter offer. However, in some cases, an autonomous entity might have the context available but will not be willing to share this context with the requesting party. In this case, policies can be defined that limit the context exchange between partners.

There are several options to implement such policies. In general, a policy can be enforced as follows. First, additional OWL restrictions and/or SWRL rules are defined to classify organizations based on the willingness to exchange context with them. This classification is then in turn used to delete certain offers. Without loss of generality, let's assume that organizations, which a provider of context wants to exchange context with are classified as belonging to the class `ExchangeOrganization`. The following generic rule is then used to delete the offers of non-trusted organizations.

```
DELETE { ?context requestedBy ?lowTrust }
WHERE {
        ?context :requestedBy ?lowTrust
        NOT EXISTS (?lowTrust
                rdf:type :ExchangeOrganization)
}
```

This thus means that the information about the WS-Agreement offer is effectively removed and will be ignored. We provide several examples of policies in Section 6.4.

6.3. *Step 3: Setup of a WS-Agreement*

Once the federation partners to which a positive reply can be sent are identified, it is possible to reply with a counter offer to the original WS-Agreement Negotiation request. To identify these counter offers, we can simply query the OWL model for all `requestedBy` properties. After the refinement process described in Section 6.2, the requests for these context are guaranteed (i) to be available in the local administrative domain and (ii) to comply with the defined refinement policies. The following SPARQL query selects those partners:

```
SELECT { ?partner ?context }
WHERE { ?context :requestedBy ?partner }
```

To each of the returned partners, a set of counter offers can be sent, which denotes that the local domain is willing to exchange the requested context. These counter offers might contain additional information, depending on the configuration of the responding partner. If the partner that originally requested the exchange of context agrees with the counter offer, a formal agreement is constructed and the counter offer will be part of the WS-Agreement document that will be exchanged between the two partners. If the partner does not agree with the counter-offer, there can be multiple subsequent negotiation steps in which the partners make counter-offers until they reach an agreement as discussed in the WS Agreement Negotiation specification. If the requesting party receives multiple counter offers it selects the best counter offer matching its requirements and sends a denial of agreement to the other partners that responded.

Once the agreement has been finalized, the partners know what context they can request from remote partners. From then on, there are no major differences between context that is available inside the local administrative domain or context that resides at other administrative domains. Therefore, the actual context exchange process is identical to the intra-domain context exchange process, detailed in Section 5.

6.4. Inter-domain context exchange in the use case

To illustrate the above described approach, Figure 8 shows how the three phase inter-domain context exchange process is executed for the use case described in Section 3. We focus on the context exchange process from the cloud provider's point of view and illustrate how he tries to find the context related to the video quality score.

First, the cloud provider broadcasts its contextual requirements (e.g., the video quality score) once to all federation partners. This is done by making a WS-Agreement Negotiation offer, which defines an obligation from the remote partner to the cloud provider that states that the video quality score must be periodically reported. In this offer, the cloud can request guarantees on this reporting. For example, it can state that the video quality score must be reported using a push-based method and their should be at least one video quality score reported every 5 seconds. Note that the above guarantees can be retrieved from the `hasUpdateFrequency` and `hasContextExchangeScheme` properties, described in Section 4. Following the WS-Agreement specification, the cloud provider can also define a reward if the above offer can be met: in this case, an augmented QoE is seen as the reward.

During setup of the agreement, the cloud provider broadcasts its contextual requirements (e.g., the video quality score) once to all federation partners. Next, the federation partners choose whether they want to exchange context based on policies. For example, the home network might define that it only wants to exchange the video quality to the cloud provider if the cloud provider is actually involved in the delivery of the video service (denoted here by the class `VideoService`) This can for example be the case if there is sensitive information linked to that video service. To express this, we can use an existential quantification, which restricts the set of companies as follows:

`Organization` **`and`** `linkedWith` **`some`** `VideoService`

In this example, the `linkedWith` property defines companies that are linked with a service. A similar example, which uses the universal quantification, is a policy that states that a provider of context only wants to exchange context with another partner if that partner does not interact with competitors. This translates into the following restriction onto the set of companies:

`Organization` **`and`** `interactsWith` **`only`** `TrustedOrganization`

The above restriction states that, if an organization interacts with another company (denoted by the `interactsWith`property), the provider of the context should also trust that organization (defined here as belonging to the class `TrustedOrganization`).

Based on the above restrictions, a semantic reasoner can classify whether organizations belong to one of the above restrictions. If they do, we can implement the described policies by defining

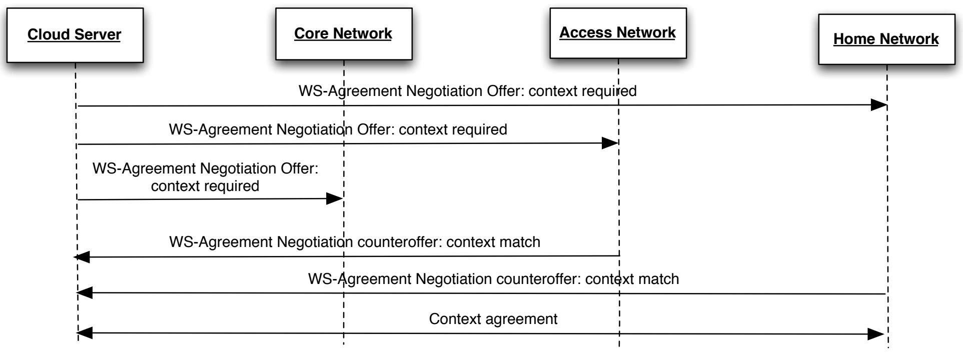

Cloud Server
Core Network
Access Network
Home Network
WS-Agreement Negotiation Offer: context required
WS-Agreement Negotiation Offer: context required
WS-Agreement Negotiation Offer: context required
WS-Agreement Negotiation counteroffer: context match
WS-Agreement Negotiation counteroffer: context match
Context agreement

Figure 8. Illustrative overview of the context exchange process between administrative domains.

new SPARQL rules and applying the above described SPARQL rules, which will delete the `requestedBy` property if the organization does not adhere to the defined policy.

It is important to note that the above policies are mere examples of how a manager can restrict policies. Other OWL features such as transitivity, symmetry, etc. can be used as well to model the relationships between federation partners.

Those partners that have this information and are willing to share it (in our case: the home network and access network) respond with a WS-Agreement Negotiation counteroffer. Finally, the cloud provider chooses to request the context from the home network and a WS-Agreement is constructed.

7. PERFORMANCE EVALUATION

We evaluate the performance of both the inter- and intra-domain context exchange processes with a particular focus on scalability. More specifically, we characterize the overhead in required bandwidth and reasoning time required by both processes as a function of the complexity of the managed environment (e.g., in terms of network size and complexity of the control loops). To do this, a prototype was implemented of both the inter-domain and intra-domain context exchange process. Both prototypes used the Jena framework [*] as ontology library. In the intra-domain context exchange process, the SPARQL rules described in Section 5 were either modeled through the SPIN library or through Jena Rules. SPIN is a W3C member submission for modeling rules in SPARQL. As discussed in Section 5, SPIN was used in two modes: with and without ECA-based rules. Jena Rules is a rule language that is part of the Jena framework, which also allows constructing new triples based on triple matching in the antecedent of a rule. We investigate the performance of the three approaches. For the inter-domain context exchange process, Pellet [*] was used as reasoner. All experiments were executed on a 1.8GHz Intel Core i7 machine with 4GB of RAM. Each test was repeated 100 times: we present average values and their standard deviation values.

To investigate the scalability of the proposed approaches, we applied it both to the use case discussed in Section 3 and performed a use case agnostic study by focusing on the scalability. In the latter case, the complexity of the RDF and OWL model was artificially increased either by increasing its size or by increasing the expressivity of the model (in case of the OWL model). We discuss these two experimentation scenarios in more detail in the remainder of this section.

7.1. *Obtained bandwidth and delay gain for the multimedia delivery scenario*

To evaluate the performance of the context exchange process, we modeled a cloud environment scenario as described in Section 3. We focus on the cloud infrastructure's root entity, and discuss

[*]Apache Jena - http://jena.apache.org/
[*]Pellet: OWL 2 Reasoner for Java - http://clarkparsia.com/pellet/

Table III. Overview of the available context types and their granularity. Context types marked with a (*) correspond with context types that are only requested when the service quality information of the corresponding connection indicates a drop in quality.

Name	Aggregated	Local
Service quality information	N/A	1 per connection
Client CPU information	N/A	1 per connection (*)
Client Memory consumption	N/A	1 per connection (*)
Network load	1 per partner	1 per router (*)
Throughput	1 per partner	1 per router (*)
Packet loss	1 per partner	1 per router (*)
Jitter	1 per partner	1 per router (*)
Delay	1 per partner	1 per router (*)

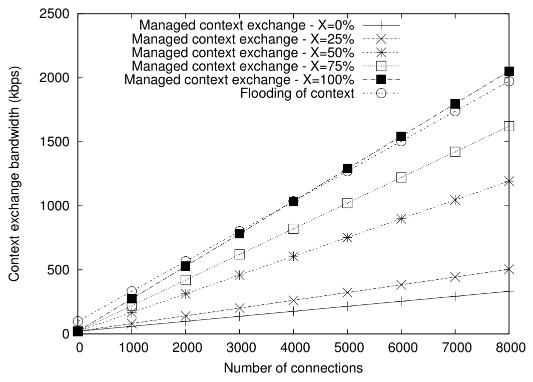


Figure 9. Bandwidth required for exchanging context in the context exchange process compared with the flooding of the federation with all context information. Different values of X, denoting the number of connections with a quality drop, are investigated. Unless all connections experience a quality drop, the context exchange process requires less bandwidth.

its context exchange process. Table III provides an overview of the different context types that are available in this model. As shown, each context type, except the types related to information at the client, has both an aggregated view (i.e., an average over a federation partner) and a local view. We assume that every context type instantiation provides an updated value of its contextual status every second. We configured the context exchange process as discussed in Section 3: the cloud provider's autonomous entity only requests the aggregated view of all context type instantiations and the service quality information of every connection it manages. Only when a drop in the service quality information is detected, local views of certain context types is requested as well. In total, 28 different rules were used to manage the scenario described in Section 3. Note that this use case corresponds with a moderate managed environment, as only 8 different context types are investigated. In practice, the available context can easily be a factor 10 higher and more. However, we describe this use case for the sake of clarity and simplicity in describing its behavior.

We compared the bandwidth that is required for exchanging context in the context exchange process with a situation where all context is broadcasted across the entire federation. For this comparison, we assumed that every context update requires 20 bytes (including metadata such as location and timestamp of the update) and that there is an overhead of sending a context update message of 150 bytes. Figure 9 shows the required bandwidth as a function of the number of connections for five distinct configurations: one corresponding with the flooding of the network with all context information and the others corresponding with the proposed context exchange process for a varying X. In this case, X denotes the share of connections that experience a quality drop and therefore require additional context information to be fetched from other partners. As shown, the required bandwidth for the context exchange scales linearly for all approaches as a function of the number of connections. This is obvious as each additional managed connection,

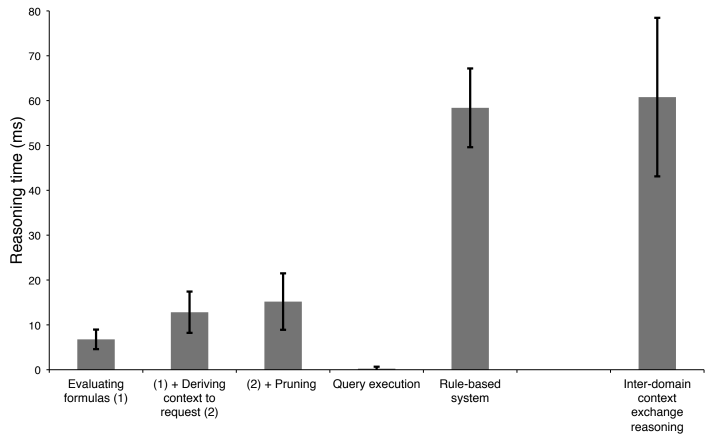


Figure 10. Time needed for performing the reasoning of each of the steps in the intra-domain context exchange process. As a comparison, the time needed to fire the rules in the actual control loop and the one time setup of the context exchange contract (as part of the inter-domain context exchange) is shown as well. We distinguish between the reasoning times during the intra-domain on-line phase (first 5 values) and the inter-domain off-line phase (last value).

requires additional quality information to be able to manage the connection. There is however an important difference in the amount of bandwidth required, even for this moderate sized use case. When flooding the federation with all context approximately 1.9Mbps is required in this use case for managing 8,000 connections as opposed to only 410kbps for the context exchange process with 10% of the connections suffering from a quality drop. In practice, the context exchange process will almost always require less bandwidth as it only requests the context that is actually needed. As shown in Figure 9, only when all connections suffer from a quality drop (i.e., X=100%), the required bandwidth is higher than the one required by flooding the network with context. This is because (i) we assumed an overhead of 150 bytes in sending a contextual update and (ii) the context exchange process will also request all knowledge in the end as all connections suffer from a quality drop. Hence, the increased overhead adds up to the required bandwidth.

Although the context exchange process results in a decrease in bandwidth, there will be an increase in reasoning time as the context exchange process will need to match the state of the dataset in order to generate the contextual requirements on-line. This is illustrated in Figure 10, which illustrates the time required of each step of the context-exchange process. Note that we used the same experimental setup, consisting of 8 different context types and 28 different RIF rules, as described above. As a reference, the time required for firing the rules in the rule-based control loop of the use case and the set up of the context contract in the inter-domain context exchange process is also shown. The firing of the rules was modeled using the Drools rule-based system platform [*]. As shown, the time required for executing the queries (i.e., to match the context type instantiations with the `requestContext` properties) is negligible as it only takes 0.43 ms. The complete reasoning time required to determine which context to request takes approximately 15.62 ms of which the first two steps both take approximately 6 ms and the context pruning only requires approximately 3 ms. In comparison, the firing of the rules for this particular use case takes on average 58.40 ms, while the one time setup of the context contract requires 60.78 ms.

The above results show that there is an overhead in reasoning time of applying the context exchange process but that this overhead is limited. Furthermore, given the large differences in required bandwidth, the flooding of the network requires a non negligible overhead in transferring the context as well. This overhead adds up to the total delay of requesting context and should also be taken into account. This is illustrated in Figure 11, which shows the total delay, consisting of inferring the contextual requirements and transferring the inferred context, as a function of the

[*]Drools - JBoss Community - http://www.jboss.org/drools/

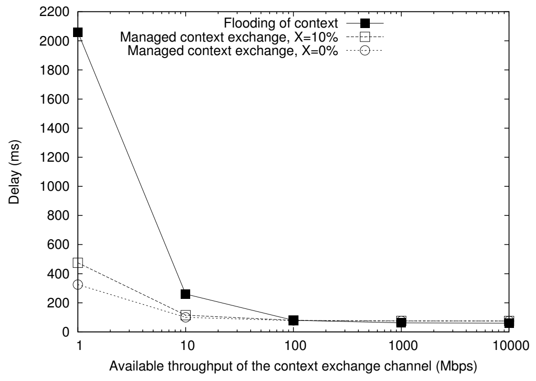
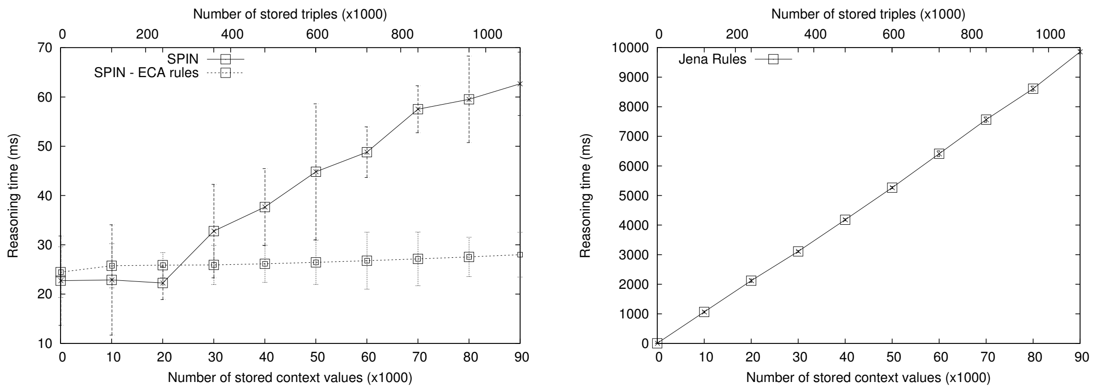


Figure 11. Total delay observed for exchanging context for managing 8,000 connections as a function of the throughput available for exchanging context. Only for very high throughput values, flooding the federation with context outperforms a managed context exchange process in terms of delay.



(a) SPIN (b) Jena Rules

Figure 12. Reasoning time required for the SPIN and Jena Rules type of reasoning in the intra-domain context exchange as a function of the number of stored context values. SPIN clearly outperforms the Jena Rules approach.

throughput available for exchanging the context. We used the values for 8,000 connections as shown in Figure 9. As illustrated, flooding the federation takes considerably more time to request and transfer the context, especially when the available throughput for exchanging context is limited. As the capacity increases, the differences become less significant. Only when 100 Mbps or more is available for exchanging context, the time required for exchanging context becomes negligible and the differences in reasoning time become the dominant factor.

7.2. *Detailed scalability study*

The results discussed above showed that there is a clear benefit in applying the context exchange process to the use case as discussed in Section 3. In this section, we carry out a scalability study of both the inter-domain and intra-domain context exchange process.

7.2.1. Intra-domain context exchange The performance of the intra-domain context exchange is mainly influenced by (i) the number of context values that need to be stored in the RDF dataset and (ii) the number of RIF rules that are used in the control loop. Figure 12 illustrates the impact of an increasing amount of stored context values on the reasoning time, both for the SPIN (Figure 12a) and Jena Rules (Figure 12b) approach. As shown, as the amount of stored context increases, the number of triples increases, the matching of the triples becomes more complex and the reasoning

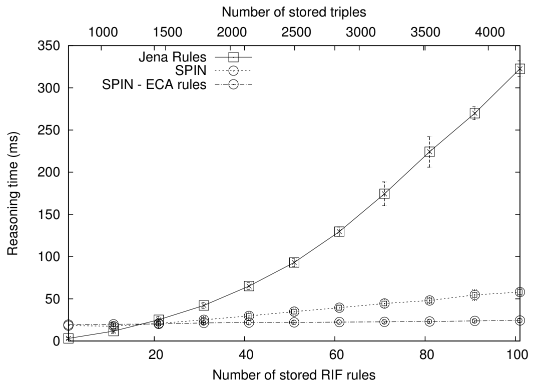


Figure 13. Reasoning time required for both SPIN and Jena Rules as a function of the number of stored RIF rules.

time increases if no ECA rules are used. There is however a significant difference between the SPIN and Jena approaches: inferring the contextual requirements with 50,000 context values stored (and consequently 600,000 triples stored) takes 44.82 ms with the SPIN approach and only 26.44 ms with the SPIN ECA approach compared to 5.26 seconds with the Jena Rules. The SPIN ECA approach is only very limitedly influenced by an increasing amount of stored context in the model. This is because the ECA-based rules allow investigating only a subset of the model and therefore avoid that all stored context needs to be evaluated. On the other hand, the Jena Rules' complexity clearly increases more as a function of the model size. Therefore, only the SPIN approach is actually deployable in an on-line setting where new contextual requirements can be generated each time the context changes and an update occurs to the RDF dataset.

The impact of the number of RIF rules of the control loop is the second factor that can influence the intra-domain context exchange. A higher number of RIF rules will not only increase the size of the RDF model, it will also influence the triggering of more SPARQL rules as the truth value of every formula of every rule needs to be investigated for the SPIN and Jena approach. Figure 13 illustrates the impact of an increasing number of RIF rules on the reasoning time. As expected, the reasoning time increases as a function of the number of RIF rules. Evaluating the contextual requirement of 100 rules takes 58.00 ms and 322.61 ms for SPIN and Jena Rules, respectively. Again, we observe that SPIN, mostly outperforms the Jena Rules approach. Only for a limited number of stored RIF rules (i.e., 10 RIF rules and lower) the Jena Rules approach performs slightly better than the SPIN approach. SPIN obviously has a higher startup overhead but this overhead quickly becomes negligible as the complexity of the model increases as Jena Rules obviously scales worse than SPIN. Moreover, the SPIN ECA approach only experiences a very small increase in reasoning time as the number of RIF rules increases. This is because, while the number of RIF rules might increase, the amount of rules that are actually investigated when using ECA-based rules, does not increase at the same pace.

7.3. Inter-domain context exchange

In this section, we investigate how the reasoning time, required to set up a context exchange contract as part of the inter-domain context exchange, scales as a function of the number of federation partners modeled in the ontology. As more federation partners are added to the model, there will be more relationships between the partners and the reasoning on their status as discussed in Section 6 will be more complex. To evaluate the scalability of the inter-domain context exchange, we investigated two types of OWL models that vary in their featured expressivity. One ontology corresponds with the OWL 2 EL language profile, while the other is an OWL 2 DL compliant ontology. OWL 2 DL is one of the most expressive technologies that are still decidable. It is a decidable fragment of first order predicate logic, with some decidable extensions that go beyond first order logic. However this large expressivity comes with a cost of a theoretical increase

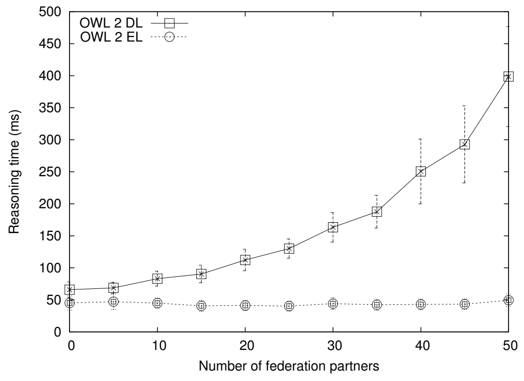


Figure 14. Reasoning time required for two different ontologies, differing in expressivity, as a function of the number of partners in the federation.

in reasoning time. The OWL 2 EL language profile is a subset of OWL 2 and is specifically designed for performing ontology consistency, class expression subsumption, and instance checking in polynomial time. It supports existential quantification to a class expression but not universal quantification.

Figure 14 illustrates the reasoning time required for both ontology versions as a function of the number of federation partners. In this experiment, the number of OWL restrictions, used to specify policies, was fixed at 10. As shown, the reasoning time scales exponentially for the OWL 2 DL ontology. This can be expected as increasing the number of federation partners, leads to a linear increase of the model size and a fully expressive OWL ontology is known to have an exponential scalability as the model increases. In this case, the performance of the OWL 2 EL ontology however, almost has no impact on an increase in the number of federation partners. This is because the OWL 2 EL language profile is specifically optimized towards large scale ontologies and the increase in the model size caused by adding 50 federation partners is still limited. Hence, there is a trade-off between defining more expressivity in the model and requiring a faster reasoning time. The OWL 2 EL language profile does not support constructs such as functional properties and disjunction of classes. Therefore, it features only a limited expressivity with the gain of an increased reasoning time.

In the previous experiment, the number of OWL restrictions was fixed at 10 and the increase in reasoning time was caused by an increase in model size. In this experiment, we increase the complexity of the model instead. Figure 15 shows the reasoning time as a function of the number of OWL restrictions and SWRL rules. Note that, in our approach, OWL restrictions and SWRL rules can be seen as two alternative techniques to model the policies. The number of federation partners was set to 10. As shown, the trends of Figure 14 are confirmed: as the complexity of the model increases, the OWL 2 DL approach experiences an exponential increase in the reasoning time. Both SWRL rules and the OWL 2 EL ontology have a more linear dependency with the reasoning time. An ontology with 100 OWL restrictions or SWRL rules leads to a reasoning time of 521.84 and 1110.32 ms, respectively. Note that this still corresponds with acceptable values in terms of reasoning time, given the large complexity of the problem domain linked with the approach.

Despite their differences, the ontology versions still obtain decent reasoning times for a complex and large scale administrative domain. As the setup of a context contract occurs only once, the timing requirements are important but less strict than in the intra-domain context exchange. The results show that even a fully compliant OWL 2 DL ontology, hence with the largest expressivity possible, can be used as the basis for the inter-domain context exchange. Inferring the context exchange contract of a federation of 50 partners, which already corresponds to a very large problem domain, only takes 398.76 ms. This value is still acceptable during the setup of a contract as the complete and automated negotiation can typically take a few seconds.

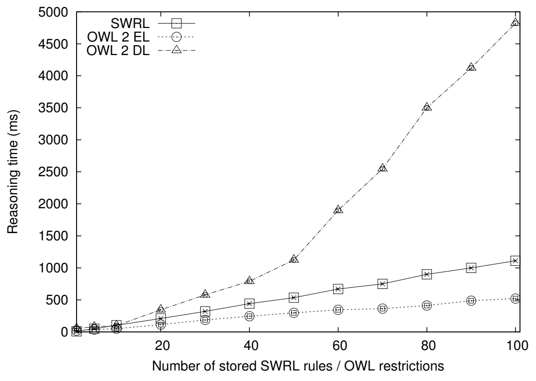


Figure 15. Reasoning time required as a function of the number of OWL restrictions and SWRL rules.

8. RELATED WORK

In this section, we discuss relevant work in the area of context exchange between remote entities and defining formal agreements inside a federation.

8.1. *Context exchange between remote entities*

In information systems, there is an increased attention towards context retrieval and dissemination research. Entities in an information system are typically overwhelmed with context and generate context of their own as well. Disseminating this context in a scalable and effective way is key in designing a well performant information system. A survey of the current challenges in designing context retrieval systems is presented by Tamine-Lechani *et al.* [15]. They particularly focus on the retrieval mechanisms effectiveness when matched by sources such as the user's search background and environment and argue that intelligent or cognitive approaches can enable high retrieval precision and allow a focus on domain-specific tasks. As the benefits of an effective context dissemination are broad and generic, it has been applied to many problem domains in information systems. For example, Bikakis *et al.* [16] present motivating scenarios for designing a context exchange and retrieval system for ambient intelligence environments. Furthermore, they present several distributed reasoning approaches for handling and resolving conflicts in contradictory context. Specifically to the field of network management, context conflicts are less of an issue but the scalability and retrieval of the context very much is. Contextual data such as configuration management data is typically stored distributed in high performant databases (e.g., the CMDBf solution from the DMTF [17]). The dissemination of aggregated monitoring information and context is necessary in order to guarantee the continuous satisfaction of federation-wide goals. The publish-subscribe paradigm is well suited to offer these functionalities. It allows interested parties to subscribe to specific types of events. When an event that matches the subscription is published, it is routed accordingly. Additionally, to ensure inter-domain understanding and interoperability, the exchanged information should be semantically annotated. This is also argued by Jurisica *et al.* [18]: they survey the use of ontologies for knowledge management in information systems and conclude that ontologies are needed to attach meaning to knowledge management systems.

The publish-subscribe paradigm has been successfully applied to the dissemination of semantic information in large-scale networked environments under the banner of *knowledge based networking* (KBN) [19]. It is an extension of content based networking (CBN) [3], which involves the forwarding of events across a network based on subscription filters based on the semantics of the (meta-)data of the event's contents. KBN extends this and states that the semantics of messages play an important part in the matching of publications to subscriptions. To this end, the Sienna publish-subscribe system, originally devised for CBN, was extended with more expressive semantics for the specification of subscriptions [4] to satisfy the KBN vision. Messages in the original Siena take the form of a set of typed attributes (i.e., name-value pairs). Filters specify constraints on the values of those attributes. The filtering process is thus purely based on the syntactical form of messages. Siena additionally supports patterns, which allows matching on combinations of messages. Carzaniga *et al.* [3] additionally propose a set of efficient and scalable routing strategies to forward messages from publishers to interested subscribers. The KBN extension [4], proposed by Keeney *et al.*, adds limited support for semantic messages and filters. Specifically, three new attribute types are added: ontological properties, concepts and individuals. Through basic ontological reasoning, filtering can be done based on semantic equivalence, super- and subconcept relationships, and property relationships. The JITIK framework presented by Brena *et al.* [20] takes a similar approach: intelligent agents form a federation and by way of delegation an agent can become responsible for specific management tasks such as the distribution of context. Ontologies are distributed across the agents: each agent has a common ontology with local additions to support its specific management tasks. Similar to our approach, ontologies are used to interpret the context that is exchanged semantically. Throughout the years, several other context exchange frameworks have been proposed. They have varying degrees of semantics and inferencing power, ranging from simple RDF graph matching [21, 22], to full-fledged OWL [23] and SWRL-based reasoning [7]. However, it is difficult to find a good balance between expressive and inferencing capabilities on one hand, and routing performance and scalability on the other. Increasing the expressiveness of the messages and filters, will generally lead to significantly reduced scalability. In this paper, we try to find such a balance by splitting up the context exchange in two parts: inside an administrative domain RDF graph matching is performed to ensure a well-performing exchange process that is able to quickly react to changes. Between administrative domains, a combination of OWL and SWRL-based reasoning is used to introduce a larger expressivity.

This article builds further on previous work as described in [24, 25]. There, we discussed how filter rules can be automatically generated by defining the contextual requirements in an ontology. The previously proposed algorithm takes into account the dynamic requirements of autonomic management components, as well as the changing state of the managed environment. Based on these inputs it generates and adapts filter rules for use in semantic context dissemination frameworks. While we extend this work, the contributions proposed in this article are significantly different. First, the work in [24] primarily focused on the generation of the filter rules. In this article, the concept is applied more broader to context exchange: while this can be implemented with filter rules this is not a requirement. Second, while in previous work, we focused solely on an OWL-based approach, this article presents a hybrid RDF and OWL-based approach, where the type of reasoning and querying is dependent on the application domain. Therefore, we are able to strike a better balance between expressivity and performance. Third, the work in [24] was applied to a single administrative domain and thus ignored aspects such as trust relationships and necessary agreements that need to be defined between federation partners. Finally, in previous work, we assumed that the contextual requirements were already modeled in an ontology. In this article, we present an approach to automatically derive these requirements using a generic interchangeable format.

8.2. *Defining formal agreements between partners*

A federation is characterized by an agreement between the participating parties. This agreement formally stipulates the rights and obligations of the involved network domains, which usually pertain to a set of shared resources or capabilities. Additionally, the agreements should specify the revenue sharing strategy, which determines how monetary gains, if any, are split across the participants. Today, the negotiation of federal agreements between network domains is a manual and time-consuming process, where business managers, lawyers, and network operators define the business, legal and technical aspects that the participating parties must adhere to. In the Future Internet, where federations will be dynamically and automatically initiated based on changing needs and requirements, this manual process should be (at least partly) automated. We envision an agreement negotiation mechanism that is automatically executed by autonomic agents. Nevertheless, they perform these negotiations within the bounds specified by high-level business, legal and technical policies, defined by human managers and operators. This means that humans are no longer directly involved in these negotiations, but can influence and govern them through the definition of high-level policies.

Automated negotiation protocols have been most commonly proposed in the context of SLA negotiation. An SLA is a formal agreement between a service provider and its customer. It specifies the terms under which a service is delivered, such as the quality, availability, or QoS guarantees. Several standardization efforts exist for the negotiation of Web Service SLAs. The Web Service Level Agreement (WSLA) specification was first proposed by IBM in 2001 [26]. It addresses the specification, creation and monitoring of SLAs. Concretely, the SLAs specify the obligations of the service provider, in terms of IT-level service parameter guarantees (e.g., availability, response time and throughput). Additionally, WSLA specifies the measures that should be taken in case the provider fails to meet its obligations. Although WSLA is equipped to incorporate an automated negotiation protocol, the actual protocol is outside its scope. More recently, the Web Services Agreement (WS-Agreement) [27] specification was introduced by the Open Grid Forum. It is a Web Service protocol for establishing agreements between parties, and has goals similar to WSLA. The agreements themselves are specified in an XML-based language. WS-Agreement incorporates three main components; a schema for specifying agreements, a schema for specifying agreement templates and a set of operations for managing their life-cycles (i.e., creation, expiration and monitoring). In contrast to WSLA, WS-Agreement does propose its own negotiation protocol called WS-Agreement Negotiation [28]. It allows two parties to negotiate on the terms of an agreement. If a compromise is reached, the negotiation results in the creation of an actual agreement using the WS-Agreement specification. Hudert et al. [29] extended the WS-Agreement specification, adding support for multilateral, in additional to bilateral, negotiations.

The definition of formal agreements through specifications such as WS- Agreement typically only introduce a syntactical description of the rights and obligations of an agreement. Therefore, it is often hard to match different offers made in the WS-Agreement Negotiation protocol. To address this, approaches have been proposed in the past that allow for semantic matching as well. Dobson *et al.* [30] describe the need for a unified ontology that semantically describes the required QoS and SLAs available between federation partners. In our approach, the ontology is not unified but an ontology matching occurs between the different administrative domains. Furthermore, the illustrative ontology used in our approach complies with the proposed requirements by [30]. They describe an overview of existing solutions and set forward a collection of requirements for such an ontology, including the typical concepts that should be modeled. Oldham *et al.* [31] define a WS-Agreement partner selection algorithm that semantically matches the agreements between partners. By describing the WS-Agreement offered Service Level Objectives (SLOs) in an ontology, they are able to reason about the meaning of those SLOs and thus extend beyond a syntactic matching of SLOs. They can for example infer that one required SLOs is in fact a subset of the SLOs that are offered. Our solution is complementary to this approach: where the work in [31] focuses on partner selection by matching SLOs, we use select partners for the context exchange process. In our case, the selection is made based on modeled relationships between the partners such as trust. The challenge of selecting partners based on trust relationships has also been discussed by Maximilien *et al.* [32]: they present a multi-agent service selection algorithm for Web Services. The proposed algorithm uses a QoS ontology and the notion of trustworthiness of the remote partner to base the decision on. Our approach differs both in the scope of the approach (Web Services vs. partners in an autonomic federated management architecture) and application (selection based on supported functions vs. selection based on contextua requirements).

9. CONCLUSIONS

In this article, we proposed two context exchange processes for autonomic federated management nodes inside and between administrative domains, respectively. The context exchange processes differ in their applicability, featured expressivity and therefore also in their experienced scalability. To determine the context that is needed inside an administrative domain, an RDF-based approach is used. This approach uses a combination of SPARQL and SPIN to determine which context needs to be requested. The process is dependent on the status of the managed environment itself: if the context about that environment changes, the context exchange process can infer that new context needs to be requested. The proposed approach does this by iteratively (i) modeling the rules available in a rule-based system of a typical control loop of a management node, (ii) investigating the status of these rules and their impact on context that is remotely available and (iii) pruning the RDF dataset to increase the scalability.

To organize the exchange of context between management domains, an OWL-based approach is used. This approach allows defining a context exchange contract that stipulates what context can be exchanged between partners in a federation and what the benefits are. We defined a context exchange process where one partner in a federation can advertise its contextual requirements to which other partners may respond positively or negatively. We discussed how the relationships modeled between partners can be used to steer this exchange process. As an example, we showed how the trust relationships can be used in combination with policies to not exchange sensitive information with poorly trusted partners. The output of this inter-domain context exchange process is a WS-Agreement compliant contract that defines the context exchange process between partners.

Both the OWL and RDF-based approaches were extensively evaluated and both applied to an illustrative use case and to a more generic setting. The evaluation of the illustrative use case showed that, even for a moderate sized managed environment, the context exchange process requires up to a factor 8 less bandwidth than a naive approach, where all knowledge is broadcasted across the federation. As a result, context can be requested up to a factor 5 times faster, despite a small additional overhead in reasoning time. The scalability study showed that the intra-domain context exchange process is able to infer the required context in a fast and timely manner. Assessing the state of the management environment and making the requests for remotely available context can be carried out in only a few tens of milliseconds, even for a large-scale model containing 90,000 context values. Furthermore, the inter-domain context exchange process can infer a context exchange contract for a large scale federation in less than half a second, while maintaining the highest expressivity in the ontology possible. We have shown that a scalable context exchange process can be built through a combination of RDF and OWL, which is expressive enough to allow the automation of context exchange and can react timely to changes in its state, even for large-scale federations. Future work of this research consists of two directions. On the one hand, we will investigate how other management approaches besides rule-based algorithms or derivatives can be modeled using RIF statements, to further increase the scope of applicability of the approach. Moreover, we will integrate the proposed context exchange process with previous solutions that focused on the design of a semantic bus to forward context and service matchmaking to detect functionality provided by federated partners. As such, a complete semantic federated architecture can be designed.

REFERENCES

1. Feeney K, Brennan R, Keeney J, Thomas H, Lewis D, Boran A, O'Sullivan D. Enabling decentralised management through federation. *Comput. Netw.* Nov 2010; **54**(16):2825–2839, doi:10.1016/j.comnet.2010.07.006. URL http://dx.doi.org/10.1016/j.comnet.2010.07.006.
2. Famaey J, Latré S, Strassner J, De Turck F. A hierarchical approach to autonomic network management. *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, 2010; 225–232, doi: 10.1109/NOMSW.2010.5486571.
3. Carzaniga A, Rosenblum DS, Wolf AL. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems* 2001; **19**(3), doi:10.1145/380749.380767.
4. Keeney J, Roblek D, Jones D, Lewis D, O'Sullivan D. Extending siena to support more expressive and flexible subscriptions. *Second International Conference on Distributed Event-Based Systems (DEBS)*, 2008; 35–46, doi: 10.1145/1385989.1385995.
5. Andrieux A, Czajkowski K, Dan A, Keahey K, Ludwig H, Nakata T, Pruyne J, Rofrano J, Tuecke S, Xu M. Web services agreement specification (WS-Agreement) 2011. http://www.ogf.org/documents/GFD.193.pdf.
6. Strassner J. *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2003.
7. Famaey J, Latré S, Strassner J, De Turck F. Semantic context dissemination and service matchmaking in future network management. *International Journal of Network Management* 2011; **22**, doi:10.1002/nem.805. URL http://dx.doi.org/10.1002/nem.805.

8. Kalfoglou Y, Schorlemmer M. Ontology mapping: The state of the art. *The Knowledge Engineering Review* 2003; **18**(1):1–31, doi:10.1017/S0269888903000651.
9. Choi N, Song IY, Han H. A survey on ontology mapping. *ACM SIGMOD Record* 2006; **35**(3):34–41, doi: 10.1145/1168092.1168097.
10. Chawathe S, Molina HG, Hammer J, Ireland K, Papakonstantinou Y, Ullman J, Widom J. The TSIMMIS Project: Integration of Heterogeneous Information Sources. *Proceedings of the 10th Meeting of the Information Processing Society of Japan (IPSJ 1994)*, 1994; 7–18.
11. Levy AY, Rajaraman A, Ordille JJ. Querying heterogeneous information sources using source descriptions. *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1996; 251–262. URL http://dl.acm.org/citation.cfm?id=645922.673469.
12. Halevy A, Ives ZG, Suciu D, Tatarinov I. Schema mediation in peer data management systems. *In Proc. of ICDE*, 2003; 505–516.
13. Steinder M, Sethi AS. A survey of fault localization techniques in computer networks. *Science of Computer Programming* 2004; **53**(2):165 – 194, doi:http://dx.doi.org/10.1016/j.scico.2004.01.010. URL http://www.sciencedirect.com/science/article/pii/S0167642304000772, ¡ce:title¿Topics in System Administration¡/ce:title¿.
14. Kim SS, seok Seo S, Kang JM, Hong JK. Autonomic fault management based on cognitive control loops. *Network Operations and Management Symposium (NOMS), 2012 IEEE*, 2012; 1104–1110, doi:10.1109/NOMS.2012.6212036.
15. Tamine-Lechani L, Boughanem M, Daoud M. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowledge and Information Systems* 2010; **24**:1–34. URL http://dx.doi.org/10.1007/s10115-009-0231-1, 10.1007/s10115-009-0231-1.
16. Bikakis A, Antoniou G, Hasapis P. Strategies for contextual reasoning with conflicts in ambient intelligence. *Knowledge and Information Systems* 2011; **27**:45–84. URL http://dx.doi.org/10.1007/s10115-010-0293-0, 10.1007/s10115-010-0293-0.
17. Distributed Management Task Force. Configuration management database (cmdb) federation specification 2010. Document Number: DSP0252 - Version 1.0.1.
18. Jurisica I, Mylopoulos J, Yu E. Ontologies for knowledge management: An information systems perspective. *Knowledge and Information Systems* 2004; **6**:380–401. URL http://dx.doi.org/10.1007/s10115-003-0135-4, 10.1007/s10115-003-0135-4.
19. Jones D, Keeney J, Lewis D, O'Sullivan D. Knowledge-based networking. *Second International Conference on Distributed Event-Based Systems*, 2008, doi:10.1145/1385989.1386034.
20. Brena R, Aguirre J, Chesnevar C, Ramrez E, Garrido L. Knowledge and information distribution leveraged by intelligent agents. *Knowledge and Information Systems* 2007; **12**:203–227. URL http://dx.doi.org/10.1007/s10115-006-0064-0, 10.1007/s10115-006-0064-0.
21. Wang J, Jin B, Li J, Shao D. A semantic-aware publish/subscribe system with RDF patterns. *28th Annual International Computer Software and Applications Conference (COMPSAC)*, 2004; 141–146, doi:10.1109/CMPSAC.2004.1342818.
22. Petrovic M, Liu H, Jacobsen HA. G-ToPSS: Fast filtering of graph-based metadata. *14th international conference on World Wide Web (WWW)*, 2005; 539–547, doi:10.1145/1060745.1060824.
23. Li H, Jiang G. Semantic message oriented middleware for publish/subscribe networks. *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, vol. 5403, 2004; 124–133, doi:10.1117/12.548172.
24. Latré S, van der Meer S, De Turck F, Strassner J, Won-Ki Hong J. Ontological generation of filter rules for context exchange in autonomic multimedia networks. *12th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2010; 575–582, doi:10.1109/NOMS.2010.5488448.
25. Latré S, Famaey J, Strassner J, De Turck F. Automated context dissemination for autonomic collaborative networks through semantic subscription filter generation. *Journal of Network and Computer Applications* 2013; doi:http://dx.doi.org/10.1016/j.jnca.2013.01.011.
26. Ludwig H, Keller A, Dan A, King RP, Franck R. Web service level agreement (WSLA) language specification. http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf 2003.
27. Andrieux A, Czajkowski K, Dan A, Keahey K, Ludwig H, Nakata T, Pruyne J, Rofrano J, Tuecke S, Xu M. Web services agreement specification (WS-Agreement) 2011. http://www.ogf.org/documents/GFD.193.pdf.
28. Battré D, Brazier F, Clark K, Oey M, Papaspyrou A, Wieder P, Ziegler W. WS-Agreement negotiation version 1.0 2011. http://www.ogf.org/documents/GFD.192.pdf.
29. Hudert S, Ludwig H, Wirtz G. Negotiating SLAs – an approach for a generic negotiation framework for WS-Agreement. *Journal of Grid Computing* 2009; **7**(2):225–246, doi:10.1007/s10723-009-9118-3.
30. Dobson G, Sanchez-Macian A. Towards unified QoS/SLA ontologies. *Proceedings of Third International Workshop on Semantic and Dynamic Web Processes (SDWP 2006*, 2006.
31. Oldham N, Verma K, Sheth A, Hakimpour F. Semantic ws-agreement partner selection. *Proceedings of the 15th international conference on World Wide Web*, WWW '06, ACM: New York, NY, USA, 2006; 697–706, doi:10.1145/1135777.1135879. URL http://doi.acm.org/10.1145/1135777.1135879.
32. Maximilien EM, Singh MP. Toward autonomic web services trust and selection. ACM Press, 2004; 212–221.