2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems

Large-scale Context Provisioning

A Use-case for Homogenous Cloud Federation

Saad Liaquat Kiani
Faculty of Engineering and Technology
University of the West of England
Bristol, UK
saad2.liaquat@uwe.ac.uk

Ashiq Anjum, Nik Bessis, Richard Hill
School of Computing and Mathematics
University of Derby
Derby, UK
{a.anjum, n.bessis, r.hill}@derby.ac.uk

***Abstract*—The ability to seamlessly bridge *clouds* across organisational and administrative boundaries will play a vital role in establishing the utility of cloud computing for large-scale collaborative processes. Managing human and environmental contexts across geographical, network and administrative boundaries is a process that can benefit from a *federation* of cloud platforms. In the absence of a mature standard that defines access, control, management and coordination mechanisms between clouds in a federation, we explore these issues through a use-case of managing the dissemination and consumption of contextual information. The use-case is driven by the deployment of a broker-based context provisioning system, for homogenous cloud deployments that reside in different administrative domains. The discussion is driven by the aim to highlight key issues and challenges for enabling cloud federation for large-scale context provisioning, which forms the main contribution of this article.**

Keywords-component; cloud federation; context management; federated brokers

I. INTRODUCTION

Context provisioning is the communication and coordination of contextual information amongst context consuming, producing and management components in a context-aware system. Context provisioning has usually been actioned by *context brokers*, which enable context-consuming components in the system to retrieve contextual information. A number of prototype context-aware systems have been developed that showcase context-awareness in several domains, but *large-scale* context provisioning and adoption of context-aware applications and services has proved elusive so far, due to multi-faceted challenges in this area. Existing context-aware systems are not ideally placed to meet the domain challenges, and facilitate their use in the emerging ubiquitous computing scenarios. Prominent shortcomings in existing systems include, 1) using a *central* context management component e.g. a context broker, for coordinating context-awareness related functions, 2) a predominant focus upon designing for static topologies of the interacting distributed components, 3) a presumption of a single administrative domain or authority and context provisioning within a single administrative, geographic or network domain, 4) a limited support for accommodating mobility of context providing and consuming components, and, 5) a lack of standardisation with respect to simple, flexible and extensible context models, for the exchange of contextual and control information between heterogeneous actors.

In this article, we will mainly focus on the third aspect from the aforementioned shortcomings that is to say, in the domain of large-scale context provisioning: consideration of only a single administrative authority and context provisioning within a single administrative, geographic or network domain. This limitation exists in part due to the design constraints in existing context-aware systems e.g. the use of central server/broker for managing context flow amongst context consumers and providers. One of the most significant impacts of this shortcoming is that the collection, aggregation, dissemination and usage of the context-aware system is then limited to a) the context information within that authoritative entity, b) to users of that administrative domain, or c) within the geographic/network scope of that administrative boundary. This impacts the range of the context-aware systems, as mobile users move out from the geographic boundary, but also it impacts the richness, usability and the quality of context information due to the limited number of context related resources available in a certain domain. These interrelated issues will continue to limit the adoption of context-aware systems and applications in future smart environments.

To overcome these limitations, context-aware systems need to implement mechanisms that permit context information *owned* or *produced* by entities under different administrative controls to share such information across the administrative boundaries. At a technological level, this requirement translates into the coordination mechanisms between the context management components of the context-aware systems i.e. context brokers or servers forming a federation for context coordination.. Such a federated broker based coordination mechanism is in line with Weiser's vision of providing contextual information about *anything, anytime and anywhere* [1], or an infrastructure that can reliably collect, aggregate and disseminate contextual information related to a very large user base over a large scale. Cloud computing is ideally placed to provide infrastructural support for meeting this

978-0-7695-4687-2/12 $26.00 © 2012 IEEE
DOI 10.1109/CISIS.2012.161

challenge through its key characteristics of reliability, scalability, performance and cost effectiveness. These benefits can be utilized by individual organisations to deploy their context management and provisioning services in private Cloud instances. However, context-aware systems have not yet taken advantage of this recent progress in the computing arena. Moreover, the federation of disparate Cloud instances is still in early stages of conceptual and technological maturity. This article illustrates a federation of homogenous context-aware systems deployed in two separate Cloud instances. The use-case of inter-Cloud federation is presented with the help of our Context Provisioning Architecture (CPA), which is a broker based context-aware system. The focus of this discussion is limited to the administrative federation aspects; the internal details regarding performance and configuration are out of the scope of this article.

The remainder of the article is organized as follows: Section II presents the background and work related to our discussion. Section III presents an overview of the federated broker-based Context Provisioning Architecture. Finally, a discussion pertaining to the key issues and challenges in a federated Cloud-based deployment of the Context Provisioning Architecture is presented in Section IV.

II. Background And Related Work

A. Federation in Dsitributed Systems

In software systems that are designed as a monolith, the whole set of functionality is embedded in the architecture and any extension and scalability is limited to pre-specification by the designers. Future evolution of such systems, with changing needs or interaction with newer systems, is costly in terms of re-design and implementation. To overcome extension limitations, large systems are usually designed as separate components encompassing different functionalities. The unit of replacement (or re-design) in component based systems is not then the whole system, but rather individual components, and the evolution in system behaviour is achieved by either adding or further specialising components. To provide scalability, multiple instances of such systems interoperate to form a larger whole and provide the same functionality as the original individual system. This approach is termed as federation [2], where two or more similar services/systems interoperate in a scalable manner. A federation may consist of similar subsystems with matching interfaces, or may provide a mediation mechanism to accommodate and achieve interoperability between subsystems, whose interfaces may have evolved to be different than others.

A number of distributed systems utilise the concept of federation to achieve scalability. Motivating factors behind the distribution of whole functionality sets (sub-systems), and then federating them, includes variation in scope and lifetime of different services in different domains. The CORBA specification describes General Inter-ORB Protocol (GIOP) for communication between ORBs [3], which also enables ORBs provided by different vendors to communicate in a federation. DNS is also an example of a federation of servers. Clients in a geographical or network boundary are served by a limited number of DNS servers to provide name resolution of systems within the boundary. For name resolution of systems that are outside the boundary, local DNS servers collaborate with other, outside boundary servers. This federation allows scalability, fault tolerance and replication in the DNS. Various broker-based messaging systems and research prototypes also discuss broker federation e.g. ActiveMQ [4], Gryphon[5], SIENA [6], JEDI [7] and HERMES [8]. The common aim of the federation function in these systems is either geographical distribution or achievement of inter-operability between different (sub-)systems. Apache Qpid [9], an enterprise messaging system, supports a mechanism by which large messaging networks can be built using multiple brokers for the connection of disparate locations across a wide area network, using departmental brokers and bridging disjoint networks. Marsh et al. [10] have also proposed the federation of Advanced Message Queuing Protocol (AMQP) message brokers, to decentralise the Message Oriented Middleware technologies used in financial markets to help scale performance across large clusters.

Federation of brokers or servers in context-aware systems has not been demonstrated as such, though a small number of works have mentioned it in their future aims. The possibility of federating multiple active spaces together using the Gaia middleware has been documented, but such an attempt has not been realized [11]. In CoBrA, the design philosophy dictates that only one context broker is to be employed as the focus is on small indoor environments [12]. A theoretical model is presented where a team of context brokers can be deployed, but the reasoning behind that approach centres on redundancy within a single domain rather than scalability and multi-domain coverage [12].

B. The need for Inter-Cloud Federation

At present, the Cloud computing focus centres around public clouds i.e. exposing platform, infrastructure or software as services to public entities such as users and enterprises. But there is a parallel momentum developing in terms of large enterprises setting up private-cloud instances for in-house utilisation. For example, two different departments in a company may carry out computational fluid dynamics simulations and the computer-aided design of a new aircraft. The two departments may also share their cloud resources, depending on the demand, amongst each other or scaling out to public clouds if their internal capacity is exhausted. There may also be a business or functional requirement of collaboration with third party private or public cloud instances e.g. with that of a partner organisation which is designing the engine for the new aircraft. Scenarios such as these require collaboration and coordination between

public and private cloud instances. The challenge lies in establishing the semantics of this collaboration and coordination, which can involve service level agreements, translation between heterogeneous Cloud API, etc.

Another scenario, related to user-centric services, is one in which telecom service providers offer context-aware services to mobile users through their Cloud infrastructures. Saturation of revenue from telephony services is well established; telecom operators and third party service providers are increasingly dependent on data and other value added services for customer attraction, satisfaction and brand loyalty. Cloud based services are being readily marketed by telecom operators [13, 14], OEM manufacturers [15] and third party service/platform providers [16]. Context-aware services present an opportunity to these enterprises for providing relevant and pro-active services for their user-base. We envision a not so distant realisation of context-aware Clouds maintained by such enterprises that host context provisioning, data aggregation/reasoning and personalisation services for their mobile customers. Moreover, these *public* context-aware services may also be augmented with those hosted in *private* Clouds e.g. one maintained by a customer's employer for providing *business* context e.g. organisational calendars, document storage, job submission and monitoring. These use-cases point towards a need for inter-Cloud cooperation, in public-public, public-private or private-private configurations, if a *holistic* context provisioning is to be achieved.

While the idea of Cloud federation has been conceptualised recently, its standardisation and practical demonstration in real world settings is very limited. Rochwerger et al. [17], during their work in the EU FP7 Reservoir project, have elicited the requirements of inter-Cloud federation in the context of business service management. Interoperability between different Cloud instances is also specified as a key requirement by the Open Cloud Computing Interface working group [18]. Goiri et al. [19] have attempted to characterise cloud federation in terms of enhancing a Cloud provider's operating profit. Their focus lies in making the decision *when* to outsource resources to other providers (and vice-versa) rather than *how* to form a federation. Similarly, Buyya et al. [20] and Bessis et al. [21] discuss inter-Cloud cooperation in terms of only scalability under a variety of load conditions. Celesti et al. [22] discuss federation amongst Cloud instances in terms of 'hot' or 'cold' disk image migration and also state that current Cloud computing platforms are *monolithic* [23] i.e. Cloud services are based on independent, proprietary architectures. They reason that the next evolutionary stage is *the vertical supply chain,* in which Cloud providers will leverage Cloud services from other providers. Their proposed eventual stage for Cloud platforms is the *horizontal federation* in which Cloud providers will federate amongst each other to achieve economies of scale and expansion of their capabilities. Based on this categorisation of evolutionary stages, they present a federation solution based on a *Cross-Cloud Federation Manager* (CCFM) component for establishing trust contexts, asset optimisation, power saving and on-demand resource provisioning. Their wholesome approach utilises discovery, match-making and authentication agents for looking up resources in *foreign* Clouds, choosing convenient foreign Clouds for establishing federation and creating a security trust [24] amongst the federated Clouds. In contrast of Celesti et al.'s independent CCFM component, Ranjan and Buyya [25] have demonstrated a peer-to-peer approach to Cloud federation that utilises a coordinating component in each Cloud instance. Their solution, entitled Aneka-Federation, is a decentralized system that integrates numerous small scale Aneka Enterprise Cloud services and nodes, which are distributed over multiple control and enterprise domains as part of a single coordinated resource leasing abstraction.

The state of the art in Cloud federation reveals efforts aimed mostly at resource sharing between Cloud instances e.g. virtual machine migration. However, the issue of cooperating processes (e.g. context providing services working towards a common goal (context awareness for users) between different Cloud instances, has not been targeted. Given the state of the art in context-aware services and Cloud computing, we envision that multi-domain context-aware systems can benefit from federated Cloud platforms in terms of holistic and large-scale coverage for a mobile user-base. The Clouds may be public or private, and we propose that such Cloud instances can be federated together to coordinate cross-organisational boundary contextual information. With this focus, we present our Context Provisioning Architecture in the following section, before discussing the key challenges in deploying context-provisioning related Cloud federation.

III. FEDERATED BROKER-BASED CONTEXT PROVISIONING

Herein, we describe our broker-based Context Provisioning Architecture briefly and discuss its inter-broker federation (non-Cloud). Next, we present the use-case of deploying this system in separate cloud instances and discuss how inter-Cloud federation is achieved.

A. Context Provisioning Architecture

The Context Provisioning Architecture is based on the producer-consumer model in which context related services take the roles of context providers or context consumers. These basic entities are interconnected by means of context brokers that provide routing, event management, query resolution and lookup services. The following paragraphs describe these three main components of the architecture.

A Context Consumer (CxC) is a component (e.g. a context based application) that uses context data. A CxC can retrieve context information by sending a context subscription to a Context Broker (CxB) and context information is delivered if and when it is available. The Context Provider (CxP) component provides contextual

information. A CxP gathers data from a collection of sensors, network services or other relevant sources. A CxP may use various aggregation and reasoning mechanisms to infer context from raw sensor, network or other source data. A CxP provides context data only to a specific invocation or subscription, and is usually specialised in a particular context domain (e.g. location). Complex components such as reasoning engines, user-profile managers and authentication services are also modelled as context providers. A Context Broker (CxB) is the main coordinating component of the architecture. It works as a facilitator between other architectural components. Primarily the CxB has to control context flow among all attached components, which it achieves by allowing CxCs to subscribe to context information and CxPs to deliver notifications. The CxBs and CxPs are typically deployed on servers (or mobile devices in special cases, e.g. a geographic position provider), whereas CxCs normally execute on user devices.

A depiction of the core system components described above is presented in Figure 1. A number of useful applications have been developed based on this architecture. Further details of this architecture and industrial trials are described in [13, 18].

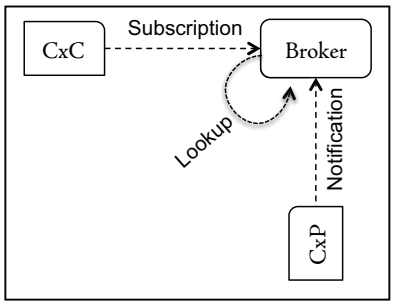

Figure 1. Basic broker-based context provisioning component interaction.

Context consumers and providers register with a broker by specifying their communication end point and the type of context they provide or require. This in turn, enables the brokering function to lookup a particular context provider that a context consumer may be interested in (e.g. based on the type of context being requested). The broker can cache recently produced context, in order to exploit the principle of locality of reference, as done routinely in Internet communications, to improve overall performance.

Contextual information, subscriptions, notification, advertisements, registration tables, etc. are all specified using an XML based language entitled ContextML. The defining principle in ContextML is that context data relates to an entity and is of a certain scope. The entity may be a user, a username, a SIP or email address etc., and scope signifies the type of context data e.g. weather, location, activity and user preferences. A specific context instance in ContextML is called a *context element* and contains the actual context data and meta-data. Context data is represented in context parameters, which are name value pairs, arrays of context parameters or in structures that are collections of context parameters and context parameter arrays.

In addition to the representation of contextual data, ContextML also contains a specification for control messages between components, subscriptions and notifications, component advertisements and routing related messages that are utilised in the overall system for coordination of context exchange. A parser, titled the ContextML Parser, has been implemented as a Java library for Java SE, EE and the Android platforms that can be used by context producing and consuming applications for the processing of contextual information and other messages encoded in ContextML. A detailed discussion about various dimensions of ContextML is presented in an earlier work [12].

B. Context Broker Federation

To reduce management and communication overheads, and achieve scalability, it is desirable to have multiple brokers in the system divided into administrative, network, geographic, contextual or load based domains. Context providers and consumers may be configured to interact only with their nearest, relevant or most convenient broker. But if context producing and consuming components only interact with a local broker, with no coordination between distributed brokers, the utility and range of the context provisioning system will be significantly impacted. Therefore, a distributed, multi-broker setup demands inter-broker federation so that context providers and consumers attached to different brokers can interact seamlessly. The brokers in the Context Provisioning Architecture work in a federation to form an overlay network of brokers (see Figure 2), which improves the scalability of the overall system [26], and provides location transparency to the local clients (CxCs and CxPs) of each broker. This federation of context brokers is achieved with a coordination model that is based on routing of context subscriptions and notifications across distributed brokers, discovery, and lookup functions, and is described in detail in our earlier work [10].

A federated broker setup is also useful for mobile context consuming and provisioning applications that may move from administrative, network or geographic domain of one broker to another. In absence of a federated broker setup, such applications would lose their subscriptions or notifications pending with the original broker, but due to broker federation, their subscriptions and notifications can still be routed to their new local broker. This mobility management mechanisms – along with related issues of disconnection of brokers – is described in detail in our earlier work [26].

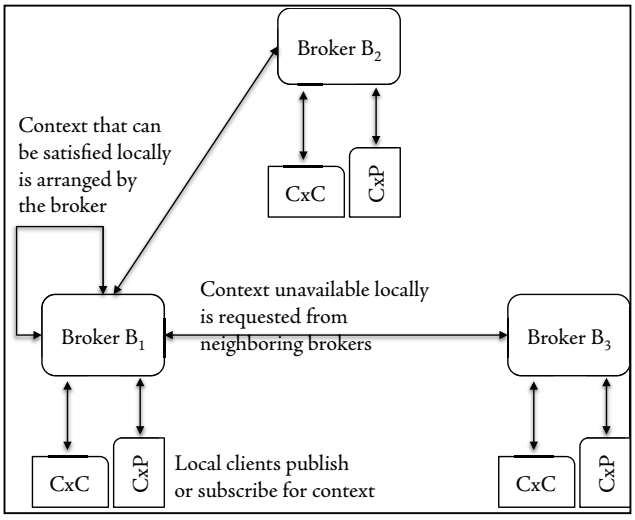

Figure 2. Simplified view of the federated brokers interaction.

1) Formation of the Inter-Broker Federation

Each CxC/CxP registers with one CxB by sending a ContextML encoded advertisement message *CxCAdv/CxPAdv*. These *client* advertisements specify the communication endpoints of the components, type (scope) of context they provide (in case of CxPs), etc. Once registered, these advertisements the CxBs add their own information (ID and communication endpoint) in these advertisements before storing them in a local client registration table. Therefore, each CxB maintains a record of its context providing and consuming clients.

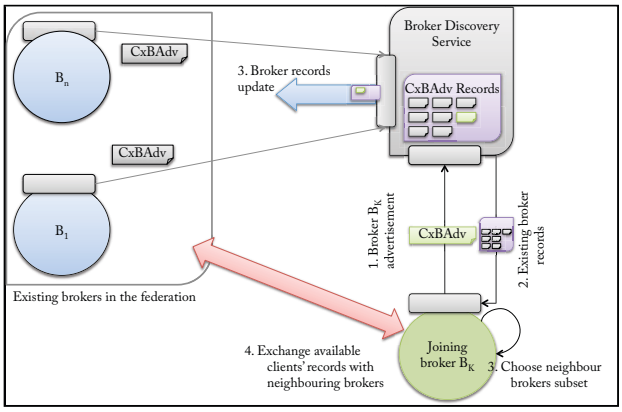

Figure 3. A new broker joining the broker federation.

The federation amongst CxBs is established with the help of a Broker Discovery and Registration Service (BDRS). A new CxB wishing to join the federation registers with the BDRS by sending a ContextML encoded advertisement message *CxBAdv*. The BDRS acknowledges the registration and sends the records of already registered CxBs to the joining CxB (see Figure 3.). After registering and receiving the list of available CxBs from the BDRS, the newly joined CxB exchanges the clients registeration tables with the existing (neighbouring) brokers. Thereafter, client registration tables are exchanged at regular intervals between the neighbouring brokers, unless a new CxP registers at one of the broker in which case that CxB will send an out of turn clients table update to its neighbouring brokers.

2) Operation of the Inter-Broker Federation

A consequence of the broker federation is that each CxP and CxC need only be aware of their local CxB. Due to the fact that federated brokers regularly exchange client registration tables, if a CxC's context related query cannot be satisfied by a CxP registered with the local CxB, it is forwarded to one of the neighbouring CxB by looking up the exchanged client registration tables. This mechanism not only provides location transparency to the client components, but also affords load scalability to the brokers and results in energy conservation in mobile devices, upon which context consuming and provisioning components may be executing [27].

The coordination model of the Context Provisioning Architecture can handle component mobility by updating client registration across the broker federation and allowing clients to change the broker with which they wish to be registered. However, this requires the components (CxPs, CxCs) executing on the mobile devices to coordinate directly with remote context brokers executing on the infrastructure. This situation is sub-optimal if a number of context consuming and providing components are executing on a mobile device, especially if the device has intermittent connectivity. Such a situation not only increases the computation and communication burden on the device, but may also compromises system security with respect to component registrations, subscriptions and notifications, thus hindering the inclusive role of modern mobile devices in context-awareness functions. To facilitate modern smart devices for an enhanced role, and enabling their active participation in functions of context-awareness, a Mobile Context Broker component is available in the Context Provisioning Architecture, that provides various context coordination and dissemination facilities, to context related applications and services executing on smart mobile devices. The Mobile Context Broker is a software component designed to execute on a mobile device as a background service that brokers the exchange of contextual information between consumers and providers, hosted both on the device and the network. Device based context providers and consumers register their presence and requirements during execution to this broker and do not have to lookup each other individually. Moreover, during periods of disconnected operation, which are still common in mobile devices and networks, these consumers and providers do not have to monitor device connectivity individually; this task is delegated to the Mobile Context Broker. Further details of the Mobile CxB component, its operation within the federation and associated benefits are presented in [27].

IV. CLOUD FEDERATION FOR CONTEXT PROVISIONING

The federated broker architecture for context provisioning, described in the previous section, can be deployed in a multi-domain environment using

conventional Java EE based servers on an IP network. The authentication of users, authorisation (e.g. access to CxPs), storage of personalised data (e.g. user profiles), etc. is carried out through a broker-based authentication mechanism. New clients (CxPs and CxCs) can be added at runtime and registered with one of the CxBs. However, the assumption in this discussion has been that the context brokers are being operated in a single administrative domain (that may span multiple network domains or geographic boundaries). When considering the deployment of the Context Provisioning Architecture in multiple Cloud instances, where each broker is hosted in a separate Cloud instance along with its associated CxPs, the assumption no longer holds and we have to consider a *single sign-on* (SSO) mechanism that can provide cross-Cloud authentication. Similarly, the non-Cloud broker based system can only be scaled manually by installing new instances of CxBs on additional servers. A Cloud-based deployment can automate this process by instantiation of additional virtual machines for hosting additional context providers, cache stores, etc. on demand. Other similar issues and our approach towards addressing them, in terms of transforming the federated broker architecture for a suitable and useful Cloud deployment, are discussed in the following subsection.

A. Key Issues and Challenges

1) Authentication and Authorisation

Client authentication and authorisation (e.g. a CxC component executing on behalf of a user) takes place through a credentials database, managed and synchronised between the context brokers. In effect, the CxPs, CxCs and CxBs execute on behalf of administrative users with particular IDs. Utilising this mechanism for a Cloud-based deployment is sub-optimal, as brokers deployed under separately administered Cloud instances may not share the same credentials database or scheme.

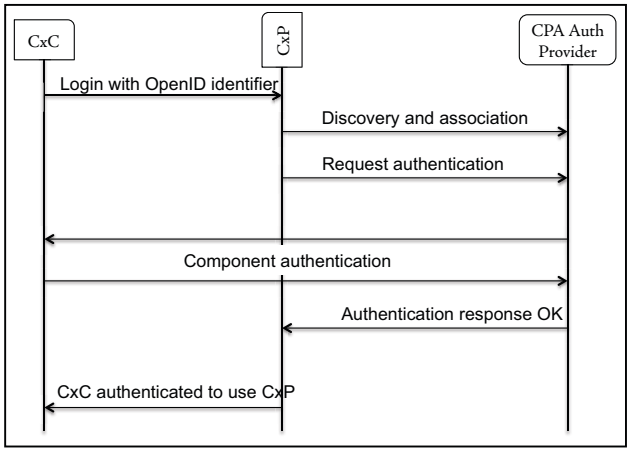

Figure 4. A example of OpenID based authentication facilitated by a Context Provisioning Authentication Provider.

An SSO scheme, such as OpenID [28], can be implemented at the context brokers such that the CxBs act as *relying parties* and verify client identities against an *identity provider*. This authentication scheme can be combined with OAuth [29] to provide a federated login mechanism, such as one available for Google account users [30]. Though this mechanism requires a one-time setup effort for configuring the context consumers and providers (and brokers for establishing inter-broker trust), it is in line with our architecture in which components execute on behalf of entities (user IDs) defined in the system.

2) Federation Establishment

Cloud instances maintained by different administrative entities are inherently individual silos, creating a need for establishment of technology and policy level agreement on coordination between hosted services and resource sharing. This requirement pertains not only to the coordination between services (CxBs and CxPs) deployed across independent (public or private) Cloud instances but also the clients (CxCs) that will use these services from different administrative, geographic or network domains. In terms of context provisioning from within a Cloud instance, an entity (e.g. a CxB) will have to make a dynamic decision about serving context in response to a CXC query from local CxPs or, in case of non-availability, contact other CxBs in cross-boundary Cloud instances to satisfy such queries. This scenario requires establishment of trust between Cloud-deployed CxBs, cross-domain authentication and authorisation for CxCs, CxPs and CxBs, and sharing service availability information across Cloud instances.

The existing implementation of the Context Provisioning Architecture uses a discovery and registration service (BDRS) for finding other brokers available for federation establishment and regularly exchanging tables of registered CxCs and CxPs. However, for a practical Cloud-based deployment, there needs to be a service level agreement between the administrative authorities for such exchange to be allowed. In addition to the brokerage of context information, service availability and that of client subscriptions and notifications, the establishment of a federation between different Cloud instances implies sharing of resources for load, administrative and functional scalability. For example, a Cloud instance managed by organisation X may contain a number of data aggregation and storage services, while one maintained by organisation Y may host computational and reasoning services that assist in synthesis of complex contextual information based on the data in organisations X's service repositories (functional scalability). Similarly, a large enterprise's authentication mechanisms may be hosted in one department's Cloud instance, which is in turn utilised by services in other, possibly geographically distributed, Cloud instances (administrative scalability).

3) Resource sharing

As discussed in Section II, recent efforts in terms of resource sharing amongst federated Cloud instances gravitate towards virtual machine migration for load balancing purposes. Within the scope of our Context Provisioning Architecture, such a migration can take place when an organisation can no longer host context

provisioning services in its current Cloud instance due to exhausted capacity and may need to instantiate such services in virtual machines in other Cloud instances in the federation. Another scenario is that of sharing resources such as cache stores. Each CxB in the Context Provisioning Architecture maintains an independent cache of context information that is provided by one of its local CxPs, irrespective of whether such context was supplied to a local CxC or one registered with a remote CxC. The utility of this caching mechanism can be improved by synchronising each broker's cache across the federation, so that any cache hits take place in a CxC's local broker instead of a broker remote to the querying CxC. This synchronisation aspect of distributed cache stores across federated Cloud instances, a major challenge in our current work, is further complicated by the presence of context brokering components in mobile devices, which we discuss in the following subsection.

4) Mobility management

The Context Provisioning Architecture caters for context consumers and providers that are mobile, not necessarily executing on smart phones but other less mobile computing devices such as notebooks. These clients register with the broker just as other static clients do, but user mobility means that these clients will move across geographic spans and network boundaries. Their involvement in the system will continue even if they are associated with the same broker after relocating (geographically or in network terms). However, context consuming and producing applications and services will mostly be concerned with the context of their geographic or network surroundings. Therefore, it is only logical for such context consumers and providers to be associated with their nearest broker, either in geographic or network terms. Moreover, clients (mobile and static) may need to disconnect from the network for other reasons such as administrative or energy conservation. Hence, remaining connected to a broker may not be continually possible. These characteristics dictate the need for a mobility management feature in the Context Provisioning Architecture. Indeed, the Context Provisioning Architecture accommodates component and device/user mobility through registration/deregistration in the normal flow of operations, and by requiring regular keep-alive advertisements from registered clients to manage situations where a client disappears without deregistering. This mobility management mechanism is described in detail in [26], including specific cases of context providers and consumers relocating from one CxB to another, disappearance of a connected component without deregistration, disconnection and reconnection of a broker in the federation, etc.

Mobility management of components in a Cloud-based deployment of the Context Provisioning Architecture does not require additional mechanisms if service level agreements and coordination mechanisms already exist to accommodate mobile components, that are not deployed in the Cloud infrastructure. The reason behind this is that non-Cloud components will only interact with the Cloud-based brokers by verifying their identities and associating, through registration, with one of the context brokers. The assumption here is that an authentication and authorisation mechanism exists (e.g. one discussed in Section IV.A.1) that can verify such mobile components to register with different brokers during mobility and utilise the context provisioning services available in the federated.

V. CONCLUSION AND FUTURE WORK

This article has presented a broker-based context-provisioning system, and discussed the aspect of broker federation for administrative, geographic and load scalability. This federation of context brokers forms the basis of our investigation into federated Cloud-instances for the provisioning of contextual information. The discussion has highlighted the requirements and possible benefits of inter-Cloud federation, for the purposes of context provisioning in large, multi-domain environments, and presented the key issues being faced in developing a Cloud-based federated deployment of the Context Provisioning Architecture.

It is our expectation that the issues highlighted through the case study of federated Clouds for context provisioning will provide an interesting and tangible platform for realising inter-Cloud federation, not only for enterprises but for end-users of context-aware services as well. The Cloud-based development of the Context Provisioning Architecture is an undergoing effort; the issues and challenges presented in this paper form the core tasks of our on-going and future work.

REFERENCES

1. Weiser, M., *The Computer for the Twenty-First Century*. Scientific American, 1991. **265**(3): p. 94-104.
2. Buschmann, F., K. Henney, and D.C. Schmidt, *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*2007: John Wiley and Sons Inc.
3. OMG, *General Inter-ORB Protocol*, in *Common Object Request Broker Architecture, Version 3.1*2008, OMG.
4. Snyder, B., D. Bosanac, and R. Davies, *ActiveMQ in Action*2010: Manning Publications.
5. Astley, M., et al. *The Gryphon Project*.
6. Carzaniga, A., *Design and Evaluation of a Wide-Area Event Notification Services.* ACM Transactions on Computer Systems, 2001. **19**(3): p. 332-383.
7. Cugola, G., E. Di Nitto, and A. Fuggetta, *The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS.* IEEE Trans. Softw. Eng., 2001. **27**: p. 827-850.
8. Pietzuch, P.R., *Hermes: A Scalable Event-based Middleware*, in *University of Cambridge PhD thesis and TR590*2004, University of Cambridge.
9. Apache Software Foundation, *Apache Qpid: Using Broker Federaton*, 2010.
10. Marsh, G., et al., *Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand*, 2009, Department of Computer Science and Engineering, The Ohio State University.
11. Roman, M., et al., *A Middleware Infrastructure for Active Spaces.* IEEE Pervasive Computing, 2002. **1**: p. 74-83.
12. Chen, H., *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*, 2004, University of Maryland, Baltimore County.

13. Vodafone. *Connecting to the Cloud: Business advantage from Cloud Services*. [Whitepaper] 2011 [cited 2012, 15 Mar]; Available from: http://www.vodafone.com/content/dam/vodafone/about/what/white_papers/connecting_tothecloud.pdf.
14. Vodafone, *Vodafone Cloud*, 2012, Google Inc.: Google play.
15. HTC. *HTC Expands Cloud Services with Dashwire Acquisition*. HTC news Releases 2011; Available from: http://www.htc.com/us/press/htc-expands-cloud-services-with-dashwire-acquisition/57.
16. Google. *Android Cloud to Device Messaging Framework*. 2010 [cited 2012, 15 Mar]; Available from: http://code.google.com/android/c2dm/.
17. Rochwerger, B., et al., *The reservoir model and architecture for open federated cloud computing.* IBM J. Res. Dev., 2009. **53**(4): p. 535-545.
18. OCCI. *Use cases and resquirements of a Cloud API*. [Specification] 2010 [cited 2012, 16 Mar]; Available from: http://www.ogf.org/documents/GFD.162.pdf.
19. Goiri, I., J. Guitart, and J. Torres. *Characterizing Cloud Federation for Enhancing Providers' Profit*. in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. 2010.
20. Buyya, R., R. Ranjan, and R.N. Calheiros, *InterCloud: utility-oriented federation of cloud computing environments for scaling of application services*, in *Proceedings of the 10th international conference on Algorithms and Architectures for Parallel Processing - Volume Part I*2010, Springer-Verlag: Busan, Korea. p. 13-31.
21. Bessis, N., et al. *Modelling Requirements for Enabling Meta-scheduling in Inter-Clouds and Inter-Enterprises*. 2011.
22. Celesti, A., et al. *Improving Virtual Machine Migration in Federated Cloud Environments*. in *Evolving Internet (INTERNET), 2010 Second International Conference on*. 2010.
23. Celesti, A., et al. *How to Enhance Cloud Architectures to Enable Cross-Federation*. in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. 2010.
24. Celesti, A., et al. *Three-Phase Cross-Cloud Federation Model: The Cloud SSO Authentication*. in *Advances in Future Internet (AFIN), 2010 Second International Conference on*. 2010.
25. Ranjan, R. and R. Buyya, *Decentralized overlay for federation of Enterprise Clouds.* Arxiv preprint arXiv:0811.2563, 2008.
26. Liaquat, S., *Federated Broker Model for Context Provisioning in Large-Scale Distributed Context-Aware Systems*, in *Faculty of Engineering and Technology*2012, University of the West of England: Bristol, UK.
27. Kiani, S.L., et al., *Context-Aware Service Utilisation in the Clouds and Energy Conservation.* Journal of Ambient Intelligence and Humanized Computing, 2012. **In press**(iUBICOM 2011 Special Issue).
28. Recordon, D. and D. Reed, *OpenID 2.0: a platform for user-centric identity management*, in *Proceedings of the second ACM workshop on Digital identity management*2006, ACM: Alexandria, Virginia, USA. p. 11-16.
29. Hammer-Lahav, E., D. Recordon, and D. Hardt, *The oauth 2.0 authorization protocol.* draft-ietf-oauth-v2-18, 2011. **8**.
30. Google. *Federated Login for Google Account Users*. 2012 [cited 2012, 21 Mar]; Available from: https://developers.google.com/accounts/docs/OpenID.