Journal of Network and Computer Applications 33 (2010) 556–568

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Enabling portability in advanced information-centric services over structured peer-to-peer systems

Jordi Pujol-Ahulló *, Pedro García-López

Universitat Rovira i Virgili, Department of Computer Science Engineering and Maths, Av. Països Catalans, 26, Tarragona, Spain

ARTICLE INFO

Article history:
Received 31 October 2009
Received in revised form
10 February 2010
Accepted 18 March 2010

Keywords:
Distributed middleware
Application portability
Peer-to-peer networks

ABSTRACT

A common factor among all the existing distributed, peer-to-peer systems is their lack of genericity. Typically, information-centric services (such as range queries) are deployed ad-hoc onto a specific peer-to-peer overlay. These kinds of solutions make them probably efficient but non-portable to other peer-to-peer infrastructures, and so the services and applications constructed over them. We do believe, instead, that a peer-to-peer-generic solution is feasible. In this paper, we tackle the genericity and portability issue specifically on structured peer-to-peer networks (SPNs).

To do so, we introduce a distributed 3-layer architecture, which abstracts applications (on top of the architecture) and the peer-to-peer network currently in use (in the bottom layer). Our middleware appears in the middle layer, which is responsible to address two major challenges: (i) supporting complex, multi-dimensional application data domains and (ii) performing efficiently for a wide variety of information-centric services in the large scale.

Broadly speaking, information-centric services are classified as data management (such as range or spatial queries) and content distribution services (like publish/subscribe), and our middleware is an umbrella for all them. Notice that data management services are based on the *pull mode* (i.e., a user *lookups* information previously *stored* in the system), whilst content distribution services obey to a *push mode* (i.e., the system delivers the information timely to users).

The benefits of our approach are clear: (i) Our middleware can be easily deployed over existing SPNs, guaranteeing the portability of a critical mass of services and end-user applications; (ii) Several services can be added to the middleware, which will facilitate the appearance of new synergies; and (iii) our middleware deals with the application data domain transparently to services and applications, including the necessary algorithms for services to be efficiently deployed into our middleware.

1. Introduction

Peer-to-peer (P2P) infrastructures have a growing remarkable role in nowadays solutions. We can see examples in several odd scenarios: in the cloud computing (Amazon uses Dynamo), in publish/subscribe services (such as Scribe on Pastry) or bibliographical sources (OverCite employs Chord). The motivation behind this P2P-enhanced technologies is to provide *scalable* and *efficient* distributed solutions, suitable for thousands or millions of users.

To make all nodes cooperate in a P2P network, lots of P2P protocols provided several ways of node inter-communication and coordination. According to the way nodes are interconnected, they are roughly classified into two categories: unstructured and structured. However, structured peer-to-peer networks (SPNs) (Aberer et al., 2003; Ratnasamy et al., 2001; Rowstron and Druschel, 2001; Stoica et al., 2001) demonstrated their better qualities against unstructured ones, such as *efficiency*, *low communication cost* or *search correctness*—i.e., if an object exists in the system, it is found. This motivated an extensive use of SPNs as infrastructure where to develop services and end-user applications.

The basic services that SPNs are able to provide (i.e., *put/get*) are not enough for up to date complex applications though. In particular, end-user applications necessitate *high level services* to successfully address the application purposes. Broadly speaking, we can organize services into two classes: *data management* and *content distribution* services. To put some examples on data management services, *k*-nearest neighbor (kNN) queries are an elementary part of document indexing applications (Tang et al., 2003) and range or window queries are the basis for geographical information services (Kovacevic et al., 2007). Conversely, publish/subscribe services for many-to-many communication, either in the topic-based (Castro et al., 2002a) or content-based (Anceaume et al., 2006) model, are services that distribute content among

* Corresponding author. Tel.: +34977558270.
E-mail addresses: jordi.pujol@urv.cat (J. Pujol-Ahulló), pedro.garcia@urv.cat (P. García-López).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.03.006

nodes. Each kind of service poses different *challenges* when developed onto P2P infrastructures, such as *efficient service provisioning* and *promptly query resolution*.

The last, but not the least, property that current services have to address is the support of *complex data domains*. Application data domains are uneven from each other. For instance, the data domain of a document indexing application is the set of keywords with which documents are described (Cacheda et al., 2005); image databases use real-valued feature vectors of length F to identify images, where each feature characterises a property from the image (Kao, 2001). Since most of the SPNs utilize a uni-dimensional numerical keyspace, this poses a *structural challenge* and must be tackled within the proposed solution.

From a system viewpoint, we can classify existing solutions into one out of these two most common approaches:

- *Constructing an ad-hoc P2P infrastructure* to support explicitly the application data domain (Banaei-Kashani and Shahabi, 2004; Bharambe et al., 2004; Harvey et al., 2003; Kovacevic et al., 2007; Zhang et al., 2004).
- *Adapting the data domain* in order to map a multi-dimensional data object to the uni-dimensional key of the SPN (Cai et al., 2003; Datta et al., 2005; Shu et al., 2005).

This data adaptation is performed by *linearization functions*, like space-filling curves (SFC) (Sagan, 1994), order-preserving hash functions (OPHF) or locality-preserving hash functions (LPHF). According to the specific context, each solution selected one kind of function to fulfill the application requisites (e.g., systems using SFCs, Shu et al., 2005, OPHF, Datta et al., 2005 or LPHF, Cai et al., 2003). However, all these solutions are designed to be efficient in the specified context, what brings a set of shortages to light:

- *Ad-hoc solutions*: The vast majority of the current P2P systems address a single problem in a given scenario, which leads to lack of *genericity* of the whole solution. In addition, most of these systems construct a solution relying on specific P2P networks.
- *Single service provisioning*: Except in very few cases (like Zhang et al., 2004), systems do not provide *multiple services* in the same solution.
- *Non-portability*: Whenever the proposed solution is tight to a given application data domain or a specific P2P network, the resulting system becomes non-portable to other applications and/or other P2P networks. Therefore, services should be re-designed and re-implemented onto additional P2P systems.
- *Maintenance*: When new services are deployed as new (overlays of) overlay networks (like Scribe on Pastry), systems suffers from duplicated signalling traffic (i.e., traffic in any of the existing overlays). This kind of traffic comes to maintain the overlay's properties, such as network connectivity or fan-out.

1.1. Contributions

Unlike previous solutions, our work aims to fill in an existing gap in the field of P2P middleware with a *generic* solution. In particular, the contributions of this paper are detailed as follows:

1. *Generic middleware*: Our solution is *flexible* and *extensible* to add new services at any time, which automatically become available to end-user applications. In addition, this middleware is *SPN-generic*, so that it can be deployed in most of the existing SPNs. This is a necessary property to ensure service portability and reuse.
2. *Application-independent*: Our middleware provides a common data structure to services and applications of any kind, which makes them independent of the underlying SPN (and of the SPN keyspace). To do so, our middleware automates transparently the adaptation of the application data domain to the SPN keyspace. Our adaptation technique is customized for each application and unifies the accessing model to nodes in the SPN.
3. *Data multi-dimensionality*: Modern applications work with complex data domains, commonly multi-dimensional. Our middleware successfully addresses this challenge, demonstrating better performance on higher dimensionalities. In addition provides the necessary algorithms to services to be successfully and efficiently deployed into our solution.
4. *Low maintenance*: Our approach relies on the rendezvous model of the SPNs to rule all supported services. Therefore, our middleware requires only a SPN to work properly. In other words, we do not construct additional overlay networks for different services.

As a proof of concept, we detail in this work three use cases, supported by different P2P networks, which will demonstrate the middleware feasibility, portability, as well as the efficiency and scalability of our approach.

The rest of article is structured as follows. Section 2 presents how existing solutions address the provisioning of new high-level information-centric services. In Section 3 we introduce our particular approach to provide a portable and scalable middleware. Our middleware is composed of three components, which are introduced in Sections 4–6. They also present the three use cases that will help to demonstrate the feasibility of our approach. In Section 7 we will evaluate the performance of the three use cases through significant simulation settings. We conclude the article in Section 8 with the final remarks.

2. Related work and background

We introduce in this section the related work from an *architectural* viewpoint, as well as from *portability and scalability* point of view.

2.1. Architecture

The designers of information-centric services must address two issues when they are deployed onto a P2P infrastructure: (i) the definition of a distributed data structure, and (ii) design of the related algorithms to perform the expected set of operations. This duality presents a trade-off that is not new, but it raises in the field of software engineering: how complex the data structure is vs how complicated the algorithms become for accessing the given data structure.

Furthermore, as mentioned earlier, the whole service is usually delivered in a *single building block*, alongside the application itself and the P2P network. In consequence, most of the systems are generically characterized by a two-layer architecture (Pujol-Ahulló and García-López, 2009). To illustrate, see Fig. 1.

The *node layer* featured the SPN state, routing capabilities and preliminary message processing. Conversely, the upper *application layer* mainly provided data storage, as well as the business logic and the distance function tool. The distance function is necessary to evaluate the closeness and to define a certain object ordering among different data objects (such as when calculating the top-K