An Environment for Decentralized, Adaptive Services: A Blueprint

Rüdiger Kapitza[1], Franz J. Hauck[2], and Hans Reiser[1]

[1]Dept. of Comp. Sciences, Informatik 4, University of Erlangen-Nürnberg, Germany
{reiser,rrkapitz}@cs.fau.de
[2]Distributed Systems Lab, University of Ulm, Germany
hauck@informatik.uni-ulm.de

Abstract. In this paper we present an environment for decentralized, adaptive services. This environment offers flexible service models based on distributed mobile objects ranging from a traditional client-server scenario to a fully peer-to-peer based approach. Furthermore the environment supports a trust-based distinction of peers and enables a trust-based usage of resources.

1 Introduction

In the last few years peer-to-peer systems have been one of the most evolving research areas. Apart form the great public demand, the utilization of peer resources is a major reason for this. In pure peer-to-peer systems every peer has almost equal responsibilities and provides resources for the whole system. This concept has certain drawbacks in context of a high number of nodes participating only for short periods of time or peers which try to attack the system. In these cases the overall system performance and service quality degrades or the system may even collapse. In contrary, traditional client-server application can cope with these problems quite smoothly but offer no possibility to utilize client-side resources. So one of the key questions is when and how a peer should participate in a distributed service.

In most peer-to-peer systems this question is often not or not sufficiently answered. Instead such systems rely on replication. However, the replication strategy is normally fixed for the whole system. It is not easily possible to adjust the number of replicas or the type of replication strategy according to changing basic conditions like the mean time of operation per node. Furthermore many peer-to-peer systems are vulnerable to attacks where a small number of entities counterfeit multiple identities so as to compromise a disproportionate share of the system [1]. Many companies, non-profit organizations and online communities would like to establish a stable, scalable service but can neither afford a traditional client-server solution, nor rely on current peer-to-peer systems because the available systems are too unreliable and vulnerable to attacks.

To fill the gap between the traditional client-sever model and a peer-to-peer based approach we propose *Decentralized, Adaptive Services*. This concept enables the usage of client-side resources in a controlled, secure fashion and provides services in a scalable, fault-tolerant manner. Decentralized, adaptive services consist of a

fragmented object [2] which is spread over a dynamic set of peers. Each part of the object might be replicated in the scope of the peer set for fault tolerance or load balancing reasons. Furthermore each part of the service is mobile and can migrate on demand of the service within the scope of the peer set. Therefore a decentralized, adaptive service can be seen as a distributed mobile agent. The peer set dynamically expands or shrinks in dependency of the participating peers. This way decentralized, adaptive services are comparable to peer-to-peer systems. But in contrary to peer-to-peer systems where each new peer has to support the system this is not necessarily the case in the context of decentralized, adaptive services. First a peer has to signal the willingness to support the service and provide information about the offered resources. Second the decentralized, adaptive service has to decide if the offering is accepted and in what kind of way the provided resources can be used in the context of the service. In fact a decentralized, adaptive service can dynamically change the internal service model from a client-sever scenario where client-side resources are simply not used to a peer-to-peer based approach by accepting only peers which offer resources and give each peer the same responsibilities for service provision. In general the peer set of a decentralized, adaptive service can be divided into at least two groups: A group of fully trusted nodes, which represent the service core, and one or more groups with peers that are untrustworthy or only partially trustworthy. The group membership depends on the service implementation and the service policy. This distinction helps to provide scalable and fault-tolerant services and use resources of partially trusted peers as needed.

The aim of the *Environment for Decentralized, Adaptive Services* (EDAS) is to provide the basic concepts and mechanisms for the development and the operation of decentralized, adaptive services. This includes mechanisms for group membership, resource monitoring and management, mobility, and mechanisms for the utilization of resources of partially trusted or untrustworthy peers.

The paper is structured as follows: The next section gives a short overview about the fragmented object model. Section 3 presents the core architecture of EDAS. In section 4 we summarize the current status of a prototype. Section 5 surveys related work and Section 6 gives some concluding remarks.

2 Fragmented Objects

In a traditional, RPC-based client-server structure, the complete functionality of an object resides on a single node. For transparent access to the object, a client instantiates a stub that handles remote invocations (Fig. 2.1 A). The stub code is usually generated automatically from an interface specification

In the fragmented object model [2], the distinction between client stubs and the server object is no longer present. From an abstract point of view, a fragmented object is a unit with unique identity, interface, behavior, and state, like in classic object-oriented design. The implementation of these properties however is not bound to a certain location, but may be distributed arbitrarily on various “fragments” (Fig. 2.1 B). Any client that wants to access the fragmented object needs a local fragment, which provides an interface identical to that of a traditional stub. However the local fragment may be specific for exactly that object. Two objects with the same interface may lead to completely different local fragments. This internal structure allows a high

degree of freedom on where the state and functionality is provided, and how the interaction between fragments is done. The internal distribution and interaction is not only transparent on the outer interface of the distributed object, but may even change dynamically at runtime, e.g., in adaptive distributed systems.

In the context of EDAS a decentralized, adaptive service is modeled as fragmented object. This offers the possibility to change the service model on demand from traditional client-server to a peer-to-peer based approach and all kind of intermediate stages by migrating and exchanging fragments.

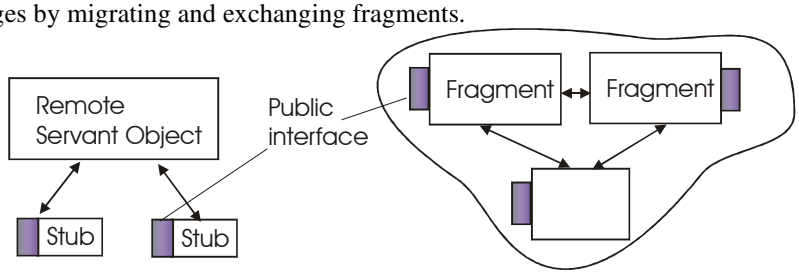

Fig. 2.1 RPC-based Client-Server Interaction vs. Fragmented Object

3 EDAS Core Components

One of the main targets of decentralized, adaptive services is to provide a concept for scalable, fault-tolerant distributed services. A decentralized, adaptive service is spread over a set of peers and combines the provided resources for service provision. Thereby the trustworthiness of the peers is taken into account. Since the peer set and the service demands many change over time the service autonomously adapts the distribution of service components. This is accomplished by migration and replication of service components as needed.

We present EDAS as a platform for decentralized, adaptive services. The platform has three major components (Fig. 3.1): A *home service,* which is provided by every peer that actively supports decentralized, adaptive services. It basically manages resources of one or more peers belonging to the same administrative domain. A *service environment,* which is spread over a set of domains that support a group of decentralized, adaptive services run by one organization or community. Finally, the *decentralized, adaptive service* which is dynamically distributed within the scope of an associated service environment.

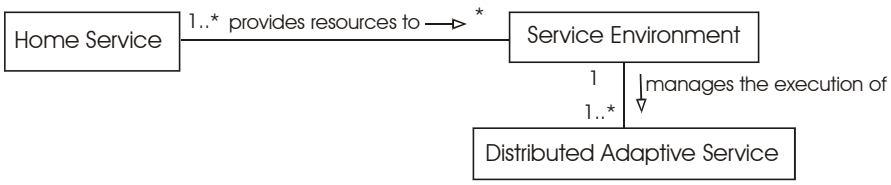

Fig. 3.1 Core Components

3.1 Home Service

The home service represents a mediator between the peers of one domain and one or more service environments each running a set of decentralized, adaptive services (Fig. 3.1). Fig. 3.2 shows three domains each running a home service which spans all peers of the respective domains. Every peer provides a set of resources. These resources are combined and monitored by the associated home service. Each domain has a manager that can assign and revoke resources through the home service to service environments. Furthermore the home service provides system information about each peer to the involved service environments and to the domain manager. This includes system load and all kinds of resource usage information but also the notification of important system events. For example, if a peer is shutdown all affected service environments are notified. In general the home service tries to provide the assigned resources to a service environment. In case of a peer shutdown the home service suggests new peers inside the domain to relocate the affected parts of a decentralized, adaptive service.

3.2 Service Environment

A server environment represents a scope of distribution for one or more decentralized, adaptive services. Normally a service environment is owned by one organization or community and managed by an individual called service manager. Such a manager can start, stop, and configure services through the interface of the service environment.

In most cases a service environment is spread over more than one administrative domain as shown in Fig. 3.2. One task of the service environment is to collect the system and resource information of the supporting administrative domains provided by the home services. Another task is to manage the migration of services or service components, based on available resources, the needs of the services, and policies provided by the service manager. The migration of service components can be necessary for various reasons like a peer shutdown, load balancing, or the growth or shrinkage of a service environment.

The expansion or shrinkage of a service environment depends on the offered resources and trustworthiness of the resource provider. Each domain manager has the possibility to offer resources to a service environment. The service manager can accept the offer and instruct the service environment to expand and use the offered resources. Furthermore the service manager can assign a trust level to the administrative domain. This rating of the resource provider allows an explicit resource usage based on the trustworthiness. Up to now the rating is based on the knowledge of the service provider but we currently evaluate how and based on what information this could be done automatically. The shrinkage of a domain can be caused by an administrative domain revoking the usage permission or simply by decision of the service administrator.

3.3 Decentralized, Adaptive Services

In EDAS a decentralized, adaptive service is represented by a fragmented object. This object expands or shrinks in the scope spanned by the associated service environment depending on the service demands and for fault-tolerance reasons. Usually every part of the object is mobile and is migrated if necessary. Each service has at least two interfaces: one for management tasks and another service specific for the end user. The management interface offers methods to start, stop, and configure services.

In the preceding section we already mentioned that each supporting domain has an assigned trust level. This level is used for a secure and fault tolerant distribution of a service. As mentioned above a decentralized, adaptive service consists of different parts. In a normal service implementation each part of a service has equal security requirements. However, if parts of the fragmented object are replicated and the changes to the replication group and the replicated data are managed by a fault tolerant algorithm, the usage of only partial trustworthy peers is possible. The service has only to ensure that the maximum number of permitted failures is never exceeded.

Another possibility is the usage of something that we call verifiable operations. For example, a service provides a set of files. These files can be placed on number of less trustworthy hosts. If a client requests a file from the service, it is transferred to the client and a checksum is provided by a part of the service residing on a fully trusted host. The client can now verify the integrity of the file with the checksum.

A third possibility is the distribution based on the self-interest of the resource provider. If the service can be structured in a way that parts of the service are only responsible for request issued by clients within a domain, then these parts should be hosted by peers of the associated domain whether they are trustworthy or not.

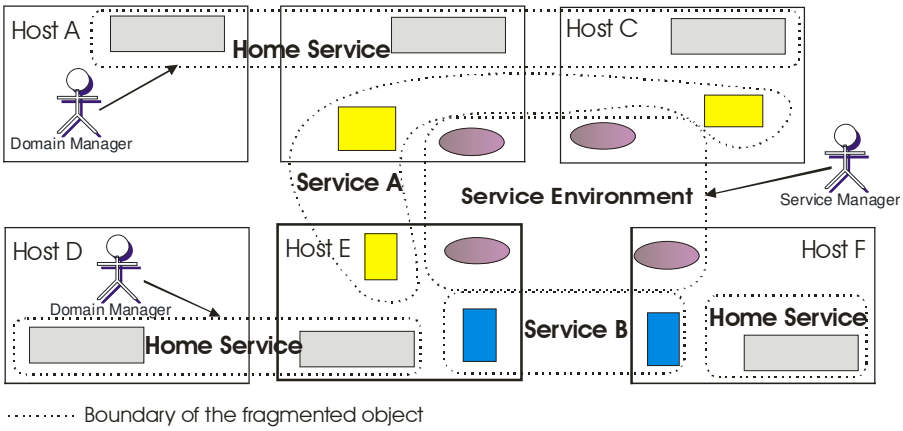

Fig. 3.2 EDAS Scenario

4 Implementation

A prototype of EADS is currently being developed in context of a project funded by the DFG. It will be based on the recently released AspectIX-ORB [3]. This ORB is CORBA [4] compliant and offers the flexibility and power of the fragmented object model. Based on former work [5] there will be an implementation of the CORBA Life-Cycle Service [6], which allows a platform independent migration in heterogeneous environments. Furthermore it is planned to implement major parts of the Fault Tolerant CORBA extension [4] inside the service environment and the home service fragment. To increase the value for a service developer, the EDAS architecture will be enhanced by two development tools. To support the replication of fragments an adaptive group communication framework [7] will be used, which is also developed in the scope of the DFG project. Furthermore another subproject aims at providing a flexible weaving toolkit. This toolkit will dramatically reduce the implementation effort to develop EDAS based services.

5 Related Work

There is a strong relation to previous mobile agent systems like Aglets [8], Mole [9] or MOA [10], but these systems where developed with a different field of application in mind. They consider an agent as a program which autonomously tries to fulfill a user specified task by migrating from one host to another and collecting data or performing computations. Therefore these systems miss a distributed object model and also flexible mechanisms for the utilization of resources provided by partially trusted hosts to support the distributed, adaptive service concept.

The CORBA Middleware offers with the CORBA Life-Cycle Service [6] and the Fault Tolerant CORBA [4] extension a basic support for decentralized, adaptive services, but also misses a distributed object model which enables the usage of client-side resources.

Grid infrastructures like the Globus-Toolkit [11] are another related research area. The main idea of grid computing is to combine a set of resources on demand to solve a special, resource consuming task. Due to this orientation, grid infrastructures lack techniques for using resources of partially trustworthy peers and the usage of client-side resources in general. Most systems also provide no support for mobile objects or mobile agent.

In the last few years a lot of peer-to-peer systems have been developed, for example CAN[12], Chord[13] or Pastry[14] to name a few. These systems construct an overlay topology and force each peer to collaborate if using the system. This has certain drawbacks if many peers participate only a short period of time or don’t behave well. The proposed concept of distributed, adaptive services offers each peer the possibility to provide resources or not. Moreover the participation of peers is controlled by a service manager.

6 Conclusions

We presented a blueprint of EDAS, an environment for decentralized, adaptive services. This environment makes two contributions. It supports flexible service models ranging from a traditional client-server scenario to a fully peer-to-peer based approach. Furthermore it provides support for scalable and fault tolerant services based on the fragmented object model and the mobility of fragments. Currently we are implementing a prototype and are looking forward to test various service implementations and resource sharing strategies. The next step will be to determine the exact interaction patterns inside the presented architecture.

7 References

[1] J. R. Douceur: "The Sybil Attack ." *Peer-to-Peer Systems: First International Workshop*, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002.

[2] F. J. Hauck, U. Becker, M. Geier, E. Meier, U. Rastofer, M. Steckermeier: "AspectIX: a Quality-Aware, Object-Based Middleware Architecture." *Proc. of the 3rd IFIP Int. Conf. on Distrib. Appl. and Interoperable Sys.* – DAIS (Krakow, Poland, Sep. 17–19, 2001). Kluver, 2001.

[3] AspectIX Research Team, Univ. of Ulm, Univ. of Erlangen-Nürnberg. *AspectIX Project Home Page.* **<http://www.aspectix.org>**

[4] Object Management Group: *The Common Object Request Broker Architectur and Specification*. Ver. 2.6, OMG Doc. formal/01-12-35, Framingham, MA, Dec. 2001.

[5] R. Kapitza, F. J. Hauck:" DLS: a CORBA service for dynamic loading of code." *Proc. of the OTM Confederated International Conferences*, Sicily, Italy,2003

[6] Object Management Group: *Life Cycle Service Specification.* Ver. 1.1, OMG Doc, formal/ 00-06-18, Framingham, MA, April 2002.

[7] Reiser: "Malicious Fault Tolerance: From Theoretical Algorithms to an Efficient Application Development Process." *Tech. Rep.* TR-I4-02-02, University of Erlangen-Nuermeberg, 2002

[8] D. Lange, M. Oshima: "Programming and Deploying Java? Mobile Agents with Aglets." Addison Wesley, 1999

[10] D. Milojicic, W. LaForge, D. Chauhan: "Mobile Objects and Agents (MOA)." Proc. *of USENIX COOTS '98*,Santa Fe, 1998

[9] M. Strasser, J. Baumann, F. Hohl: "Mole - A Java based Mobile Agent System." In: M. Mühlhäuser: (ed.), Special Issues in Object Oriented Programming. dpunkt Verlag 1997

[11] I. Foster, C. Kesselman, S. Tuecke: "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." *International J. Supercomputer Applications*, 15(3), 2001

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker: "A Scalable Content-Addressable Network." *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, August 2001

[13] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan: "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications." *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, August 2001

[14] A. Rowstron, P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems." *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November, 2001.