

# How Much Middle-Tier Do You Need?\*

Uwe Roth  
Institute of Telematics  
Bahnhofstr. 30-32  
D-54292 Trier  
Germany  
roth@ti.fhg.de

Kais Louizi  
Institute of Telematics  
Bahnhofstr. 30-32  
D-54292 Trier  
Germany  
louizi@ti.fhg.de

Ernst-Georg Haffner  
Institute of Telematics  
Bahnhofstr. 30-32  
D-54292 Trier  
Germany  
haffner@ti.fhg.de

Christoph Meinel  
Institute of Telematics  
Bahnhofstr. 30-32  
D-54292 Trier  
Germany  
meinel@ti.fhg.de

**Abstract:** Middle-tier-technologies have been realized to be the only useful way of controlling data-flow between the internet and databases in the intranet. In the last years there have been many different approaches to solve this middle-tier-problem. In this paper, we present a middle-tier-platform called the "Smart-Data-Server" and show its flexibility and applicability with a practical example and point out the main differences between this middle-tier platform and other approaches.

## Introduction

At the beginning of the internet era nobody would have expected that the idea of world-wide-connected computers could have such a huge impact on today's information-technology (IT). Most companies and authorities had to reconsider and redesign their IT-infrastructure to fulfill the new requirements. Companies realized that the new form of IT is essential to survive and continue running their business successfully. In order to reduce costs and improve service quality authorities felt the need of introducing these new technologies to their IT-infrastructure.

When companies started connecting LANs to a big WAN and hence providing an infrastructure for client/server communication they faced a big problem: "How can clients efficiently access data on servers?" Some technologies such as Perl-Scripts and Java-Servlets on Web-Servers were supposed to provide a solution to this problem but did not provide enough flexibility. The need of using a middle-tier-architecture was obvious as more powerful technologies were required to provide a layer to control data-flow between Web-Servers and the databases in the intranet. In the last years many different approaches have been made to solve this middle-tier-problem. In this paper, we present a middle-tier-platform called the "Smart-Data-Server" and show its flexibility and applicability with a practical example and point out the main differences between this middle-tier platform and other approaches.

## The Structure of the Smart Data Server (SDS)

The Structure of the SDS has been presented on the WebNet'99 [9] conference as a middle-tier platform for distributed applications but for better understanding we will introduce it shortly.

The SDS consists of three layers: the first layer is responsible for handling the requests to the server as a server-session, the second layer offers services to the components of the SDS whereas the last one contains the main functionality of the SDS in form of components. Each component can be added and removed to the server independently without affecting the other components of the SDS. The components are only allowed to communicate with the "outer-world" of the SDS via the services of the service-layer, so the SDS can be adapted to different environments only by configuring the services. The access to databases for example is encapsulated by a datastore-service. Requests towards these services are handled by special datastore-objects that can be configured by the components. The execution of the requests itself is not visible to the components that use this service. The datastore-service translates the requests to standard-SQL-statements and executes them via JDBC-connections. The datastore service does not use any vendor specific feature, which means that no specific functionality can be used. This makes it easier to switch between databases of different vendors without affecting the SDS or its components.

The components of the SDS offer services that can be accessed by clients sending requests to the SDS. The requests are well formed XML-documents [12] composed of the requested function, the parameter values and the desired result.

---

\* In Proc. "World Conference on WWW and Internet", AACE WebNet 2001, Orlando, Florida, 2001, pp 1052-1056

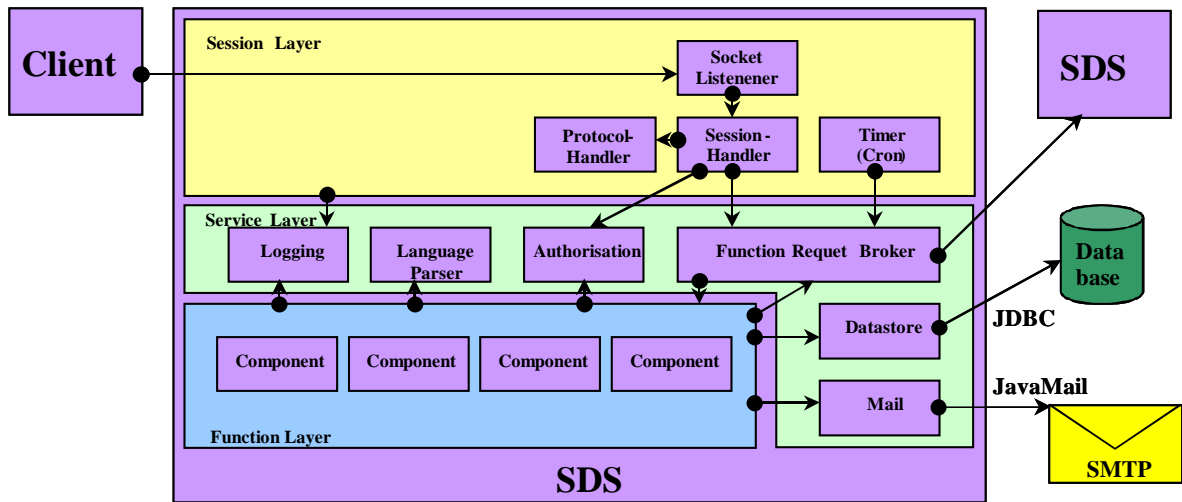


Fig.1: The internal Structure of the Smart-Data-Server.

The main advantage of using the SDS-platform as a middle-tier architecture lies in the short development time. The SDS, as a framework for building multi-tier applications, enables developers to build their applications in a very short period of time as it offers several services that do not have to be developed from the scratch.

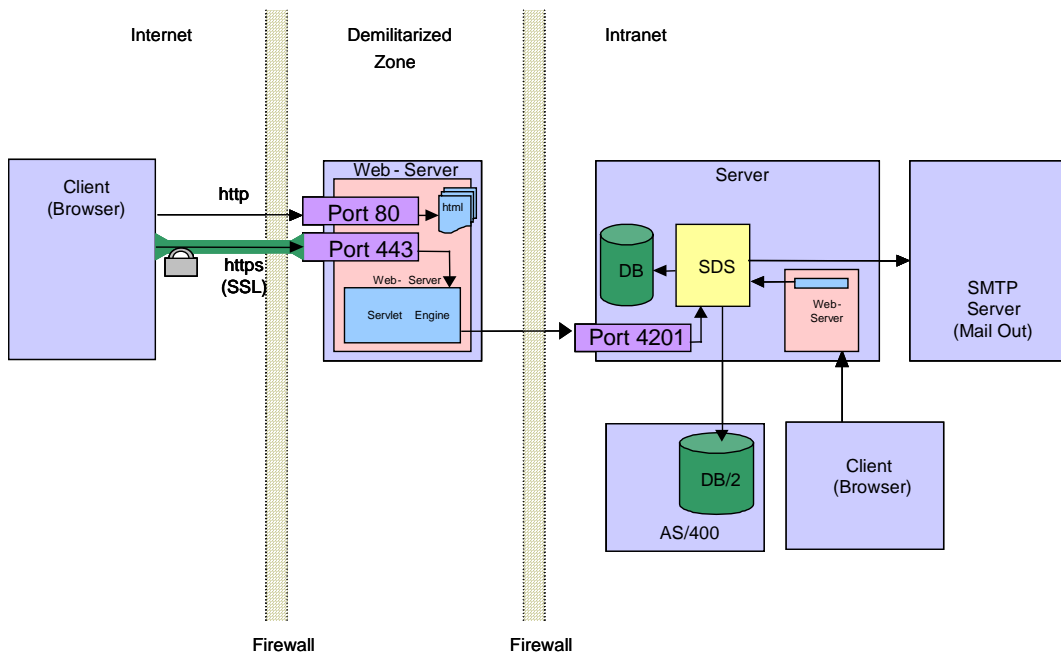


Fig. 2: First Plan.

## Practical Example

The project we present in the following was a cooperation between the Institute of Telematics and the ministry of education in the state of Rhineland-Palatinate, Germany. Our main task consisted in developing an internet application to allow all the teachers in the state of Rhineland-Palatinate to extend their contracts.

Fig. 2 shows an architectural overview of the first concept.

We identified two groups of users, on the one side the teachers who send the applications and on the other side the officials in charge that are responsible for processing these applications. Due to the big number of teachers (more than 8000) we could not make any assumptions on the platforms they might be using. So we decided to

provide a HTML front-end for our application that can be accessed by a conventional web browser running on any platform. The officials in charge can also access the system via web browsers.

A teacher fills out an online form and sends it to a servlet running on the web server, which will on its turn transfer the data to the SDS. When filling the form the web browser interacts with the SDS via servlets to provide the user with relevant information and check his/her input. During this interaction only non-critical data (such as school names, possible contract-models, etc ...) is transferred to the internet. Critical data is kept safely within the intranet.

At an advanced stage of the project we have been informed that the architecture we had designed could not be realized in its actual form due to security policy violation. In fact, the firewall between the demilitarized zone and the intranet was only opened for mail protocols and hence would not have allowed the servlets to communicate with the SDS, even if secure protocol had been used.

We had to reconsider the communication between the intranet and the internet to develop a new concept.

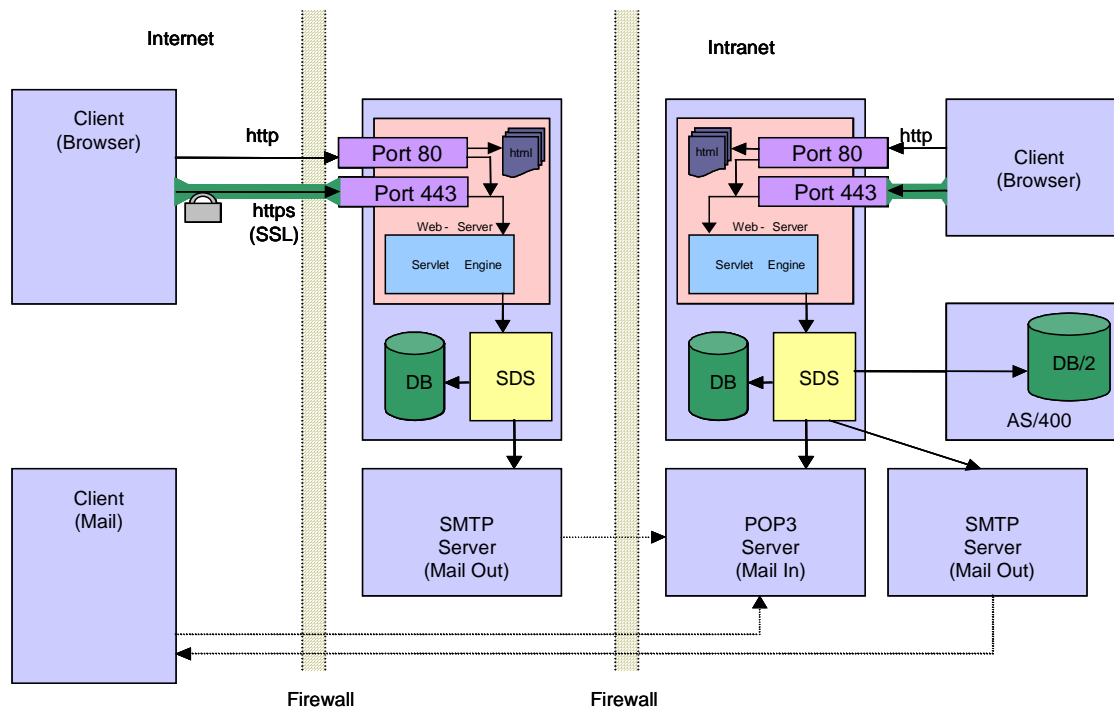


Fig. 3: Second Plan.

Fig 3 shows the redesigned architecture. The main difference to the first concept is the utilization of another SDS, referred to as “Internet-SDS”, in the demilitarized zone. Instead of sending their requests to the Intranet-SDS, the servlets are now communicating with the Internet-SDS in the demilitarized zone. The structure of requests remained unchanged, only their destination had to be changed. The Internet-SDS generates an e-mail containing the requested data and sends it to the POP3 server in the intranet. Once the mail arrives at the intranet the Intranet-SDS can access it and retrieve the relevant application data. Notice that the new concept uses the e-mail protocol to fulfill the goal of storing application data in an intranet database and conform to the security policy at the same time. The flexibility offered by the SDS played a major role in reducing the changes in design and especially in implementation.

The request-syntax for addressing the “insert application”-components of the Internet-SDS and the Intranet-SDS is the same but the actions performed by the components are different. The functionality of the Internet-SDS-component consists of creating an (encrypted) email with all the application-data and sending it to an email-account owned by the Intranet-SDS. The functionality of the equivalent Intranet-SDS-Component is the insertion of the application-data into the local database. Another component had to be added to the Intranet-SDS. This component is responsible for a cyclic check of the email-account, reading, encrypting the relevant emails and requesting the “insert-application”-component.

The information about whether a request sent to the SDS should be handled by a local component or should be routed to a second SDS is held by the function request broker of the SDS and can easily be changed in the SDS-configuration-file. In case of a change of the security policy in the future that will lead to the firewall being

opened for dedicated requests from the Internet-SDS to the Intranet-SDS, a single change of the configuration-file will route the “insert-application”-requests directly to the Intranet-SDS. The use of email to transfer data from the internet to the intranet will then no longer be used.

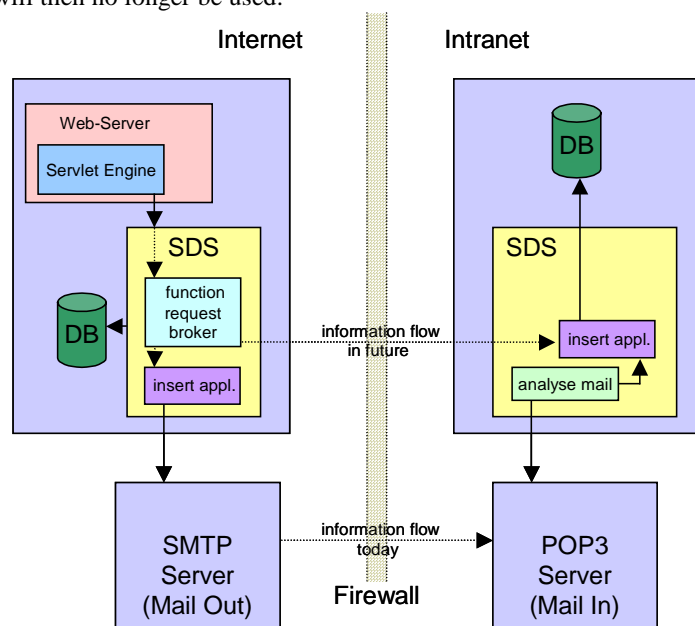


Fig. 4: Information exchange between the Internet-SDS and the Intranet-SDS.

## Related Work

The SDS described in this paper represents an attempt to address the problem of building multi-tier systems for the internet. As one of the most challenging problems of today's computer industry this question has been addressed by the leading companies and organizations. Today, the most widely used and deployed distributed technologies for building multi-tier systems are the Common Object Request Broker Architecture (CORBA) [6] and Enterprise JavaBeans (EJB) [2][7]. CORBA provides a platform independent component-based distributed computing infrastructure. EJB is the Java-based equivalent technology to CORBA. It also defines a component-oriented framework for developing distributed applications. The question that raises now is: How can SDS be compared to CORBA or EJB? Can SDS be considered as an alternative to these technologies?

CORBA and EJB address the same problem as the SDS, which is developing web multi-tier applications. Another similarity is that the SDS, like CORBA or EJB, is component based. As a developer you can implement components and add them to the SDS by referencing them in configuration files. Components have to implement a specific interface to be able to communicate with the SDS.

Now let us have a closer look at the differences. Whereas CORBA or EJB provide a general infrastructure for building applications, the SDS provides a high-level framework users only need to customize to build a web application. Users are provided with a base application that can be easily extended to a complex one. Developing a web application based on CORBA means most of the time starting from scratch. Moreover, the SDS is equipped with services that can be used without any programming effort. Another major difference between the SDS and CORBA lies in the target problem field. CORBA addresses a much larger problem field, namely client/server programming in general. It also specifies vertical services that can be used when developing applications for specific fields. The SDS was designed with web applications in mind and it was meant to be used to develop web multi-tier applications. The new CORBA specification (CORBA 3.0) focuses more on internet applications. Finally, the SDS supports only pure java applications whereas CORBA applications can be developed using different programming languages such as C++ or COBOL.

Hence, the SDS can be thought of as web application server rather than an enabling technology such as CORBA or EJB. Certainly we could think about developing a CORBA-SDS or an EJB-SDS that takes advantage of these standard technologies.

## Summary and future work

In this paper we presented a multi-tier application developed using the Smart Data Server, a middle-tier-platform [10][11].

First we depicted briefly the SDS architecture. We highlighted the most important components and explained how the SDS can be used for developing multi-tier systems. Then we focused on an implementation of a practical application using the SDS. The developed system consists of two major components that are meant to be used by two user groups. The first component provides the teachers in the state of Rhineland-Palatinate, Germany, with a channel to apply for extending their contracts. The second component allows officials in charge to process the receive applications.

We explained how the SDS allowed us to redesign the system in a short time to be able to fulfill system requirements as well as to conform to the security policies of the ministry of education of the state of Rheinland-Palatinate.

SDS can be thought of as an application server for developing web applications with a specific component model. This component model is based on similar concepts as the CORBA or EJB component model. We think that redesigning SDS in the future to use a standard technology such as CORBA can improve SDS. The CORBA 3 standard addresses several problems related to development of internet applications such as firewalls or asynchronous method calls. In case of redesigning SDS it would be very useful so to use CORBA 3 and take advantage of its new services.

## References:

- [1] Irvine et. al., *Hypertext Transfer Protocol -- HTTP/1.1*, Available from World Wide Web: <<http://www.w3.org/Protocols/rfc2616/rfc2616.html>>
- [2] Sun Microsystems, *Enterprise Java Beans, 2.0 Specification*, Available from World Wide Web: <<http://java.sun.com/products/ejb/2.0.html>>
- [3] Sun Microsystems, *Remote Method Invocation (RMI)*, Available from World Wide Web: <<ftp://ftp.java.sun.com/docs/j2se1.3/rmi-spec-1.3.pdf>>
- [4] Robert Orfali, Dan Harkey, Jeri Edwards, *Client/Server Survival Guide*, John Wiley & Sons, 3<sup>rd</sup> Edition, 1999
- [5] Jason Hunter, William Crawford, *Java Servlet Programming*, O'Reilly & Associates, 2<sup>nd</sup> Edition, 2001
- [6] Michi Henning, Steve Vinoski, *Advanced CORBA Programming with C++*, Addison-Wesley Pub Co, 1999
- [7] Richard Monson-Haefel, *Enterprise Java Beans*, O'Reilly & Associates, 2<sup>nd</sup> Edition, 2000
- [8] Jim Farley, *Java Distribute Computing*, O'Reilly & Associates, 1998
- [9] Uwe Roth, Ernst-Georg Haffner, Thomas Engel, Christoph Meinel *An Approach to Distributed Functionality - The Smart Data Server*, Proceedings of the World Conference on the WWW and Internet, AACE WebNet'99, Honolulu, Hawaii, USA, 1999, pp 931-1539
- [10] Uwe Roth, Ernst-Georg Haffner, Thomas Engel, Christoph Meinel *The Smart Data Server – A New Kind of Middle-Tier*, Proceedings of the IASTED International Conference IMSA 99, Nassau, Bahamas, 1999, pp 361-365
- [11] Uwe Roth, Thomas Engel, Christoph Meinel *Improving the Quality of Information Flow with the Smart Data Server*, Proceedings of the International on Internet Computing IC'2000, Las Vegas, Nevada, USA, 2000, pp 353-357
- [12] W3C, *Extensible Markup Language (XML) 1.0 (Second Edition)*, Available from World Wide Web: <<http://www.w3.org/TR/2000/REC-xml-20001006>>