Interoperability Using APPC

Debajyoti Mukhopadhyay

Bellcore, 444 Hoes Lane, Piscataway, New Jersey 08854
debm@ctt.bellcore.com

Abstract

The complex and competitive business world of today needs to access data for various operations from different systems placed at different geographical locations. In order to fulfil this need, one should have a reliable distributed computing environment. Various components of that environment may be supplied by different vendors. This means that the computing environment not only needs to be distributed but also requires interoperability. Today, Interoperability is no more just an idea but a reality. There is also a growing need to support interactions among various systems in a dialog mode. Integrating distributed systems can only help to achieve the goal of developing a reliable distributed computing environment. In this paper, a conceptual framework for an architecture is described in conjunction with Advanced Program-to-Program Communications LU 6.2 protocol to handle that challenge. This architecture discusses the required contracting services for integrating distributed systems. This contracting service has three major components: the contract interaction services, the contract support services, and the communications infrastructure services.

1. Introduction

Many enterprises today rely on complex and pervasive software applications to run their businesses. Increasingly, businesses are integrated concerns requiring integrated operations. Applications supporting the different business operations are developed using a variety of technologies and vendor hardware and software. This variety offers businesses a selection of applications that best fit their needs. However, this variety can result in an environment in which applications are stand alone islands and operations are fragmented. Such a situation can no longer serve business needs. An application must be able to interoperate with other business applications of various vintages and suppliers, and that are deployed on a variety of computing environments. Interoperability is *the ability to interconnect applications regardless of their suppliers and vintages, to provide access to these applications and to corporate data by any authorized user, and to maintain that interconnection and access over changes in suppliers and vintages.*

The Bellcore OSCA™ architecture is an architecture designed to enable and enhance interoperability among applications.[1] The OSCA architecture's goals and concerns are related to the interoperability needs of a business. The OSCA architecture delineates generic requirements and principles for building blocks 1) that are interoperable and operable; 2) that offer open access to all of their functionality to all authorized users and other building blocks; 3) that allow rapid changes and improvements in functionality by separating concerns between business operations, corporate data management, and user interaction processes *so that only the affected building blocks need to be replaced, not the entire product;* 4) that manage redundant private and corporate data to provide a single consistent view of business information; 5) that enable selection and seamless integration of various pieces of software that are developed in-house by the users, reused, or acquired from multiple vendors; 6) and that allow deployment of software products across a variety of hardware/software platforms by requiring adherence to appropriate industry and/or national/international standards. Fundamental to the OSCA architecture is the notion of separation of concerns. The OSCA architecture requires that business application functionality be separated (grouped) into "layers" (not to be confused with any OSI layers) or domains: a corporate data layer, a processing layer, and a user layer. A layer is the union of all functionality defined as either corporate data functionality, processing functionality, or user functionality. The corporate data layer stewards corporate data and provides application data management functionality to support all create, retrieval, update and delete operations of corporate data in a semantically consistent manner. Seman-

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0479...$1.50

tic consistency is taken in a broad sense to encompass business-meaningful integrity. The corporate data layer also supports redundancy management and ad-hoc retrieval. The user layer provides functionality to support human users and business application functions that directly interact with human users. The processing layer provides functionality to support business operations and management, such as telecommunications call processing control and non-interactive process control. The software that implements the functionality in these layers is partitioned into building blocks. Interfaces between building blocks must meet certain criteria. Such interfaces are termed "contracts."

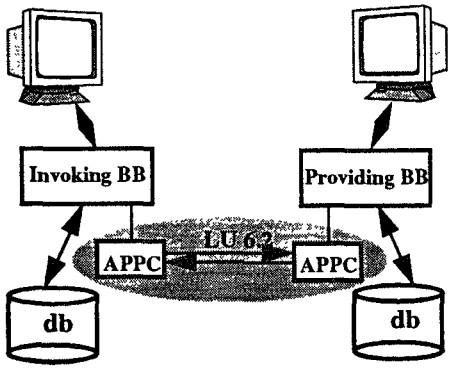

Figure 1 . APPC Inter-Building Block Communication

This document deals with the inter-building block interactions in a dialog paradigm only. See Figure 1. The participating building blocks can be a combination of user layer, processing layer or data layer building blocks. The environment described for illustrative purposes here is MVS™, though the same concept can very well be applied to other environments of the reader's choosing.[2][3][4][5][6][7][8][9]

2. Conceptual Framework

A conceptual framework with the help of a data flow diagram is shown in Figure 2 to give an idea of how communication should take place between two MVS systems using LU 6.2 protocol in the framework of the OSCA architecture. In a typical case, the TP#1 passes a contract name, and constraints on contract attributes to the contract interaction services (CIS) to invoke a contract. CIS calls the contract support services (CSS) with the contract name and constraints for discovering a contract instance.

Within CSS, the contract trader calls directory services. The directory services return all available contract instances that match with the contract name to the contract trader.The contract trader selects a particular contract instance, say TP#2 in this case, which meets the constraints on contract attributes, and the logical unit at the remote site (LU#2). The security server gets the contract access point (CAP) of the invoking site, and the user id, contract name, and any other relevant security information. The security server returns the information indicating whether access is allowed to the remote site. The CSS passes all this information to the CIS.

The CIS now issues an ALLOCATE verb with the contract instance, security information, etc. to establish a conversation within a session between the local and remote systems. Issuing of the ALLOCATE verb results in passing the LU#2, TP#2 and other information to the local Virtual Telecommunications Access Method (VTAM™).

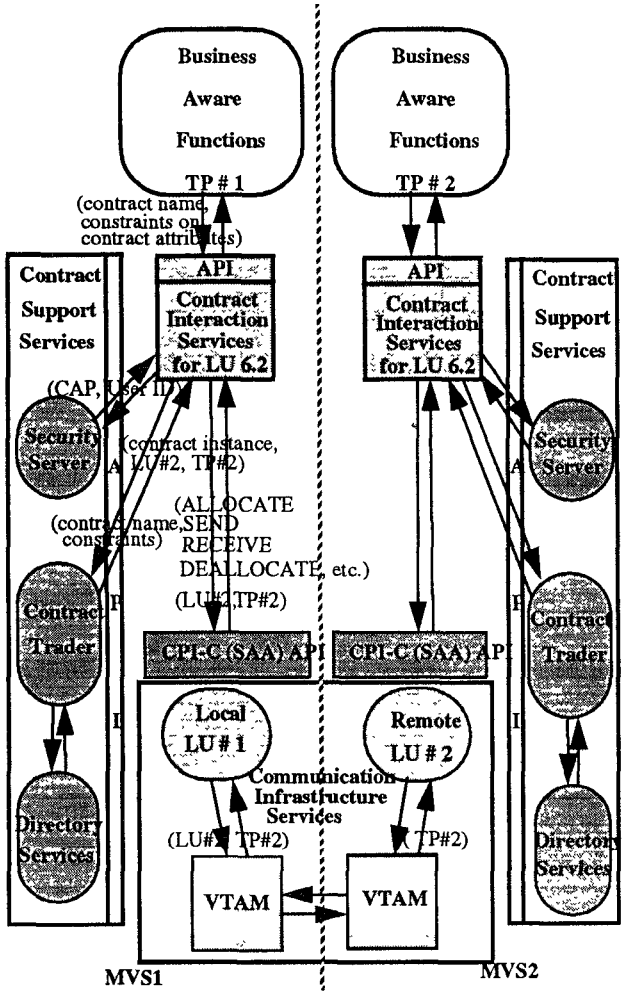

Figure 2 . Conceptual Framework of Architecture in a Distributed Environment

The MVS1 VTAM, which is part of the communication infrastructure services (CInfS), looks into its resource definition table and finds the address of the remote VTAM. The CInfS sends the LU#2, TP#2 and other information across the network to the remote VTAM. The remote (receiving/inbound) VTAM looks into its own resource definition table and resolves the physical address of the LU#2. It passes the TP#2 to the remote logical unit (LU#2). This results in establishing a conversation within a session.[10][11][12][13][14][15][16][17][18]

3. Contract Interaction Services

The application calls the contract interaction services at the invoking site to perform necessary actions so that the contract is processed at the invoked site (remote site). A set of verbs are utilized in LU 6.2 to establish a session/conversation, to carry on a conversation, and to discontinue a conversation.[19][20] There should be an understanding between the contract interaction services at two or more different systems to process those verbs (e.g., ALLOCATE, SEND, RECEIVE, CONFIRM, etc.) to establish a session/conversation, and to disconnect a conversation.

4. Contract Support Services

The contract support services are usually used by the contract interaction services and the application. They include: contract trading, security, logging, and encoding/decoding. It should be mentioned here that these services are necessary in a distributed environment.

4.1. Contract Trading & Directory Services

The contract trader is responsible for resolving a global contract name and constraints on instance attributes to an address of a particular provider instance of a contract. The contract trading service helps to hide the distributed nature of the OSCA environment by enabling the distributed processing environment in which the building blocks can trade and invoke contracts. In an OSCA environment, the building blocks interoperate via contracts using logical addressing. In this environment, some building blocks provide contracts and other building blocks invoke those contracts. A contract might be provided by multiple building block instances as well, such that there will be multiple instances of a contract available in the environment. Therefore, an invoking building block instance should be able to discover (search and select) an appropriate instance of a contract based on its requirements. It is also required that there be a central repository where these contracts can be registered to indicate the availability of these contracts in the system. All these aspects of contracts such as contract registration, discovery and logical addressing or address mapping are the functionalities of a contract trader.[21]

A global naming scheme is required to support the contract trading in this environment. The naming scheme should support directory capabilities for all entities (e.g., building blocks, contracts, etc.) requiring location transparency. The directory is a repository of information about entities, and the services used to access that information. The directory services provided to its users are concerned with various kinds of access to this information. The directory consists of one or more directory servers.[22]

4.2. Security

It is necessary to have some kind of security mechanism for the LUs. Before APPC applications can communicate, one must define LUs to VTAM. This is done by defining the LUs to VTAM with a VTAM APPL definition statement. In a conversation mode, the outbound TP passes a userid, password, and the security_program as parameters on an ALLOCATE request for an inbound TP on MVS. Also, there should be a security protection at the contract interaction services level. This should be done by issuing a SEND verb with the remote building block id and user id as its parameters after the ALLOCATE verb has already been used to successfully establish a session/conversation. This should be done only once for any specific dialog. The remote building block should send a reply by either confirming or rejecting the request. A security server is responsible for validating the contract access point, user id and any other relevant information for security purposes.

The API between the security server and the contract interaction services is GSS (Generic Security Service). The GSS-API defines an interface to cryptographically implemented authentication and other security services at a generic level which is independent of particular underlying mechanisms. For example, GSS-API-provided services can be implemented by secret-key technologies (e.g., Kerberos) or public-key approaches (e.g., X.509). The GSS-API is independent of the communications protocol suites with which it is employed, permitting use in a broad range of protocol environments. The GSS-API's security context construct is independent of communications protocol association constructs.[23]

5. Communications Infrastructure Services

Communication infrastructure services are responsible for delivering messages between systems. LU 6.2 requires SNA to provide these services. In SNA, it is the path control network which takes care of routing messages from node to node. However, it is not necessary to discuss the path control network here. Rather, it is worth mentioning

that the resource definitions in the local VTAM must correspond to the VTAM in the remote site. The LUs that the local site is going to access over the network must be defined in the remote VTAM.

6. Conclusions

This paper has described a conceptual framework of an architecture to realize interoperability using APPC. The major contribution of this work is the identification of the infrastructure services and a suitable framework to implement it. References to specific vendor products are strictly for illustrative purposes. No endorsement of any product or vendor is intended. Each component of the contracting services is discussed in details. The conceptual framework is substantiated with a clear description of inter-system communications. Another strong aspect of this framewok is its ability to support dialog paradigm. In summary, a practical approach offering interoperability using APPC LU 6.2 protocol in an interactive mode has been presented.

OSCA is a trademark of Bellcore, Inc.
MVS, SAA, and VTAM are trademarks of IBM Corporation.

References

[1] *The Bellcore OSCA™ Architecture*; Bellcore Technical Reference TR-STS-000915; Issue 1; October 1992.

[2] J. A. Mills & L. Ruston; *The OSCA Architecture: Enabling independent product software maintenance*; In Proc. of EUROMICRO'90 Workshop on Real Time; Copenhagen Denmark; 1990.

[3] J. A. Mills; *Interoperability of Network Data Functionality with Operations Systems Data Functionality*; In Proc. 6th World Telecommunication Forum; Geneva, Switzerland; 2(III), 1991.

[4] J. A. Mills; *An OSCA Architecture Characterization of Network Functionality and Data*; Journal of Systems Integration; 1(1), Kluwer Academic Publishers; 1991.

[5] D. Mukhopadhyay; *An Approach to Realize Interoperability Using APPC*; International Journal of Computer Communications; Butterworth-Heinemann Ltd. UK (to appear).

[6] D. Mukhopadhyay; *Managing Interoperability Using APPC*; Proc. of the IEEE First International Conference on Systems Management; Los Angeles, California; April 1993.

[7] D. Mukhopadhyay; *A Generic Retrieval System Design Supporting Interoperability*; International Journal of Information Systems; Pergamon Press, Oxford, UK; Vol. 18, No. 4, 1993 (to appear).

[8] D. Mukhopadhyay; *A Generic Information Retrieval System to Support Interoperability*; ACM SIGIR Forum; New York; Vol. 27, No. 1, Spring 1993 (to appear).

[9] D. Mukhopadhyay; *Design of a Generic Retrieval System for a Distributed Environment*; Proc. of the USING'92 International Conference on Open Systems for the Telecom Industry; Philadelphia, Pennsylvania; May 1992.

[10] *Application Development: Writing Transaction Programs for APPC/MVS*; IBM System Product 5695-047; March 1991.

[11] *An Introduction to Advanced Program-to-Program Communication (APPC)*; IBM GG24-1584-1; November 1986.

[12] *Systems Network Architecture: Transaction Programmer's Reference Manual for LU Type 6.2*; IBM GC30-3084-4; September 1991.

[13] *Planning: APPC Management*; IBM GC28-1110-0; March 1991.

[14] *APPC and CPI-C Product Implementations;* IBM GG24-3520-01; July 1991.

[15] *VTAM APPC API Scenarios and Programming Experiences;* IBM GG24-3297-00; February 1989.

[16] *Performance Considerations in an APPC/MVS Environment;* IBM GG66-3206-01; September 1991.

[17] *Distributed Function Design Considerations;* IBM GG24-3767-00; November 1991.

[18] *Communications Systems Bulletin: Advanced Architectures APPC, SNADS, DIA and DCA;* IBM GG22-9105-00; March 1985.

[19] A. Berson; *APPC: Introduction to LU 6.2;* McGraw-Hill, Inc.; 1990.

[20] J. Martin; *SNA: IBM's Networking Solution;* Prentice-Hall, Inc.; 1987.

[21] *OSCA™ Contract Trading Guidelines*; Draft Bellcore Special Report SR-STS-002278; March 1993.

[22] *OSCA™ Naming and Addressing Framework*; Draft Bellcore Special Report SR-STS-002319; March 1993.

[23] J. Linn; *Generic Security Service Application Program Interface: Internet Draft;* Digital Equipment Corporation; June 1991.