THE ARCHITECTURAL REQUIREMENTS AND INTEGRATION ANALYSES OF A DATABASE SERVER FOR OFFICE AUTOMATION *

Steven A. Demurjian, David K. Hsiao, and Roger G. Marshall

Department of Computer Science
Naval Postgraduate School
Monterey, California 93943

ABSTRACT

Office automation systems are growing both in use and in complexity. The development of a database management system for the office automation environment becomes a high priority, in order to provide an efficient and reliable way to manage the information needs of the office. Therefore, the specification of a database server for the office automation environment becomes a key area of concern. In addition to providing traditional database support, the database server must also provide new database support, so as to meet the unique and many needs of office automation environments. In this paper, we focus on the characterization and specification of a database server for the office automation environment. We also consider how such a database server can be effectively integrated into the office automation environment. We use both intuitive comparisons and queueing-theory analyses to evaluate the various integration approaches. Further, we examine an experimental database system, known as the multi-backend database system (MBDS), as a candidate for the database server in the office automation environment.

1. AN INTRODUCTION

As office automation systems (OAS) become more prevalent in the work place, the need for database support in the office automation environment (OAE) becomes a key issue. In this paper we attempt to provide the characterization of a database server for OAEs. The database server is used to provide traditional as well as new database support in the OAE. In addition, we study various approaches to the integration of the database server into an OAE. In our characterization and study of a server, we focus on the use of an experimental database system, known as the multi-backend database system (MBDS), as the server. Although MBDS may not be an ideal choice, it does serve as a benchmark for measuring the other database servers for OAEs. In the rest of this paper we examine how and why MBDS may be considered as a database server for the OAE.

More specifically, in Section 2 we discuss the architecture and characteristics of a database server for the OAE. In Section 3 we briefly describe the design and implementation of MBDS. We also analyze whether the unique design characteristics of MBDS meet the needs of the OAE. In Section 4 we present how a database server such as MBDS can be integrated into the OAE by providing an intuitive comparison of the possible design configurations. We focus on the multiple-backend architecture of MBDS and how it satisfies the architectural requirements of the OAE database server. In Section 5, we use single-device and multiple-device queueing models to formally evaluate the design configurations given in Section 4. Finally, in Section 6 we present our conclusions.

2. A CHARACTERIZATION OF AN IDEAL DATABASE SERVER

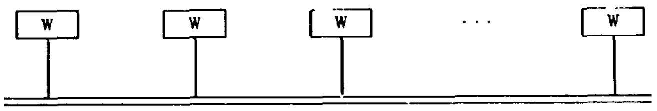
When characterizing a database server for the OAE, we focus our efforts in two directions. First, we consider the architectural requirements of the database server that will facilitate the smooth integration of the database server into the OAE. Second, we consider the necessary database system features or characteristics of the database server for the OAE. In the following two sections, we examine these two considerations for a database server in the OAE.

2.1. The Architectural Requirements

The basic structure of the OAE consists of a group of workstations connected by a local-area network (LAN). (See Figure 1, where a workstation is denoted with the letter W in a square box.) To successfully meet the needs of this environment, the database server must be integrated into the existing OAE. The integration of the database server into the OAE must be

1985 ACM 0-89791-170-9/85/1000-0121 $00.75

* The work reported herein is supported by grants from the Department of Defense STARS Program and from the Office of Naval Research and conducted at the Laboratory for Database Systems Research, Naval Postgraduate School, Monterey, CA 93943.



Local Area Network (LAN)

Figure 1. The Basic OAS

smooth and have no ill-effects on the existing OAS. If the database server runs only on a single workstation, it should be powerful enough to meet the database management needs of the present OAE. However, it may not meet the growing demands of database management of the future OAE. Thus, it seems logical that the database server should consist of, initially, a few workstations and, later, a number of workstations. With multiple workstations, the database server should reduce and distribute the database management load across the variable and multiple workstations.

Whether those workstations which make up the database server act as individual and separate database systems or cooperate to handle the database management needs of the OAE is an issue. It may not be feasible in a given OAE to distribute the database management functionality and load among different database servers on the same network, since the OAS is not a distributed database system. More likely, the OAE requires a central repository of data and programs that is maintained and accessed via a single system, so that the data and programs can be successfully shared throughout the OAE. This calls for a database system for centralized databases.

In conclusion, the overall architectural requirements of a database server for OAEs should therefore dictate a centralized database system running on variable and multiple workstations.

2.2. The Six Characteristics

There are six major characteristics of such a database server. They are software portability, software independence, auto-configurability, survivability, versatility, and performance. *Software portability* provides the database server with the ability to be accessible on a wide range of workstations. Specifically, the database server should not be restricted to a particular class of workstations and a specific type of operating systems. Instead, it should be portable across a wide range of workstations and operating systems of the OAE. If the database server is implemented on multiple workstations, the software components of the server running on the separate workstations should be sufficiently independent, so that the database server does not become inoperative when a node (i.e., either one of the software components on a workstation or a workstation itself) becomes disabled. *Software independence* among system components running on separate workstations may eliminate software and hardware interdependencies and the complexity of the database server.

When running on variable and multiple workstations, the database server should be *auto-configurable* and *reconfigurable*. When the OAE grows, (i.e., the number of workstations in the OAS increases) or shrinks (i.e., some workstations becomes disabled or removed) the database server should be able to adjust itself for the addition or loss of workstations. Such adjustment should require no new programming and no modification to the existing software. Further, it should incur no disruption of the OAE or OAS. The database server should also maintain a consistent and up-to-date copy of the database. When a node in the OAE is disabled, it is imperative that the database server still be functional, providing continuous, albeit limited, access to the remaining database. This is also the *survivability* of the database server.

The database server should also be *versatile*, providing the user with more than one way of accessing the database. In an OAE where there is a large group of individuals from diverse backgrounds and with different experiences in using database facilities, the database server should provide different database language interfaces in order to facilitate the database user with various ways of accessing the database. Finally, the database server should be a database system that is oriented towards providing a substantial level of *performance*. As the time goes by, both the use of and repository in the database server increase. To meet the growing needs of the OAE the database server must be able to expand as the OAE expands, and still maintain or increase its performance.

3. THE NEED OF A DATABASE SERVER WITH VARIABLE AND MULTIPLE BACKEND CONFIGURATIONS

3.1. The Proposed Architecture for a Database Server

We advocate that the architecture of a database server be configured with one controller and one or more backends. As shown in Figure 2, the controller and the backends are connected by a broadcast bus. When a transaction is received from a workstation, the controller broadcasts the transaction to all the backends. Each backend has a number of dedicated disk drives. Since the database is distributed across the backends, a transaction can be executed by all the backends in parallel. Each backend maintains a queue of transactions and schedules the transactions for execution independent of the other backends, in order to maximize its access operations and to minimize its idle time. Thus, different transactions can be executed concurrently. On the other hand, the controller does very little work. It is responsible for receiving and broadcast-

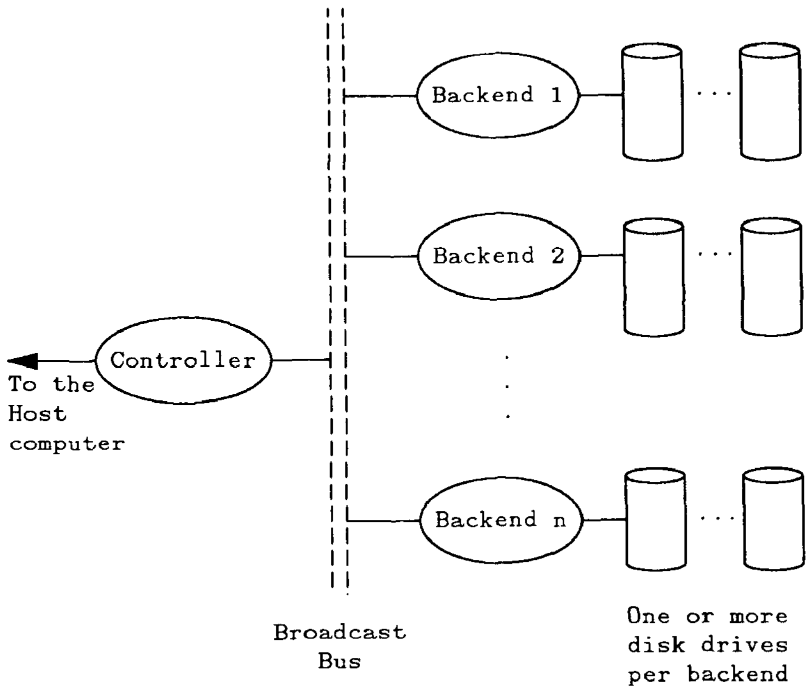


Figure 2. The Multi-Backend Database System (MBDS)

ing transactions, routing results, and assisting the backends in the insertion of new data. By minimizing the work of the controller, we are attempting to reduce the chances that the controller may become the bottleneck in the system when the number of backends is increased. The backends do all the database operations. Just how this architecture may have the six characteristics of an ideal database server will be expounded in the following sections by way of an experimental database system which also exhibits an architectural configuration similar to the one depicted in Figure 2.

3.2. The Multi-Backend Database System (MBDS) as a Database Server

To provide a centralized database system, MBDS uses one or more identical workstations and their disk systems as database backends and a workstation as the database controller to interface with multiple, similar or dissimilar workstations or mainframes. We refer to these workstations and mainframes as ***hosts***. User access to the centralized database is therefore accomplished through a host which in turn communicates with the controller. Multiple backends are configured in parallel. The database is distributed across all of the backends. The database management functions are replicated at each backend, i.e., all backends have identical software and hardware. They, of course, have accesses to different portions of the database.

There are some key issues to be explored in considering MBDS for OAEs. The current implementation of MBDS uses microprocessor-based workstations, Winchester-type disks and an Ethernet-like broadcast bus. There are a number of reasons for preferring microprocessor-based workstations over the traditional minicomputers. First, the 32-bit microprocesser is quickly attaining a reputation as a dependable, versatile and fast CPU, approaching the speed and performance of the minicomputers of five years ago. Second, the microprocessor-based workstation is cost-effective. This is important when considering that MBDS requires a minimum of two such workstations. It also implies that MBDS can be expanded with relative ease and minimal cost by the addition of backend microprocessor-based workstations.

The placement of the user interface is also affected by the use of microprocessor-based workstations. The user interface provides access to MBDS and is run from either a separate host, or as part of the backend controller. When the user interface is on a separate host, the interface interacts with the controller via a bus. In either case, the use of a similar (with respect to the controller and backend hardware) microprocessor-based workstation for the user interface increases the compatibility and the maintainability (with respect to the hardware maintenance and costs) of the database system.

The final major issue involves the ability of MBDS to support multiple data model/language interfaces to the multi-backend database system. These multiple model/language interfaces allow the user to access MBDS using the relational model/SQL language, the hierarchical model/DL/1 language, the entity relationship model/Daplex language, or the network model/CODASYL language. These interfaces are also running on either a separate host or the backend controller; and, as such, the issues concerning the user interface also apply here. For a brief introduction to the multi-backend database system MBDS and the multi-lingual database system MLDS, the reader may refer to [4].

3.3. Six Characteristics of MBDS for an Effective Role in the OAE

Regardless of the integration approach that is chosen, MBDS exhibits certain characteristics that are desirable in the OAE. These characteristics include the software portability of the MBDS code, the software independence of the backend code, the auto-configurability and reconfigurability of MBDS on account of its use of identical workstations and replicated software, the survivability of the system resulting from the use of duplicated directory data, the versatility of system due to the ability of MBDS to support multiple language interfaces, and the performance capabilities of the system as a result of its parallel configuration and round-robin data placement. Each of these topics is briefly examined in the following paragraphs.

3.3.1. Software Portability

The MBDS processes, i.e., the controller processes, the backend processes, and the interface process, are all written in the C programming language. C was chosen as the programming language for MBDS because of its portability and its reputation as a good systems programming language. We estimate that the code of MBDS is about ninety-five percent portable, consisting of 13,000 lines of C code. The five percent of system-dependent code involves the inter-process message-passing code on both the controller and the backends, the inter-computer message passing code for the communications processes, and the disk I/O routines for the record processing process. Thus, the majority of the code is portable. In fact, some of the implementation development for MBDS takes place on a VAX-11/780 running the Unix operating system, where we are able to take advantage of the C-tools provided by Unix. Thus, we feel that we have designed a relatively portable database system that can run on a wide range of the 32-bit microcomputers on the market today, e.g., the DEC MicroVAX and the Sun Workstation.

3.3.2. Software Independence

In examining the software independence issue, we focus on the backend processes. The elegance of MBDS is in that the backend software of one backend is identical to the backend software of another backend. For logical reasons, the directory data, used by each backend when processing requests, is nevertheless duplicated at every backend. However, the directory data is usually a small percentage of the non-directory data. Furthermore, the only sharing of information by the backends occurs in one phase of the directory search. Otherwise, the directory management, the concurrency control, and the record processing processes are independent of each other. So, when a new backend is configured into the system, the software present on one backend is simply replicated on the new backend. Additionally, the directory data, duplicated at an existing backend, is loaded into the new backend. When bringing a new backend into MBDS, we must also decide on whether to rearrange the non-directory data. On the one hand, we can redistribute all of the non-directory data across the disk systems of every backend. This involves the reloading of data. On the other hand, we can simply leave the data undisturbed, and load only the new data on the new backend. The choice is left to the discretion of the database administrator.

3.3.3. Auto-Configurability

One of the most convenient features of MBDS is the ability to automatically configure and reconfigure the system with ease. When starting the system for the first time, the database administrator simply specifies, using an interface, the number of backends in the system. MBDS then configures itself by notifying the controller and backend processes the number of backends on the system. Using this unique feature, MBDS can be reconfigured when a backend becomes inoperable. In such a situation, MBDS is configured with one less backend. Conversely, when a new backend is added to the system, the system can be configured with one more backend easily.

3.3.4. Survivability

MBDS contains only one copy of the non-directory database. When the database is loaded, it is distributed evenly across all backends' disk systems. However, the directory data, which contains index and cluster information on all data in the database, is duplicated in every backend. The distributed directory data, coupled with the software independence and reconfigurability of MBDS, offers an increased survivability of the database system in the OAE. If a backend or backends become inoperable, the system is still usable. While a backend is inoperable, a log of transactions that modified both the directory and the non-directory data is kept. When the backend is reconfigured into MBDS, the log is run for the purpose of updating the directory and other data. Although portions of the non-directory data become inaccessible with the inoperable backends, MBDS can still access and retrieve the rest of the data. Incomplete data is better than no data, provided that the user is informed of the situation.

3.3.5. Versatility

One of the biggest advantages of having MBDS as part of an OAS is the ability of MBDS to provide support for multiple data models (and therefore data languages) through the use of multiple language-based interfaces. In the OAE, where users are from diverse backgrounds, such a utility is a unique feature in a database management system. In fact, the language interfaces can be tailored by the workstation. One workstation could have a SQL interface, another a DL/I interface, a third a Daplex interface, and yet another a CODASYL interface. By tailoring the language interfaces by workstation, the software required for each interface process could be reduced. Conversely, with a wide range of language interfaces available at every workstation, the workstation becomes more accessible to a wide range of users.

3.3.6. Performance

The performance capabilities of any DBMS are important in an OAE, since the DBMS tends to serve as a repository of all the permanent data and programs of the OAE. As the repository becomes large and the database activities increase, the DBMS as a database server may become the performance bottleneck. However, MBDS is specifically designed to provide for capacity growth and performance enhancement. The performance metric of major concern is the response time of a request. The *response time* of a request is the time between the initial issuance of the request and the receipt of the final results for the request. MBDS has two original design goals. First, if the database capacity is fixed and the number of backends is increased, then the response time per request reduces proportionately. For example, if a request had a response time of 60 seconds when there was one backend, the same request should have a response time of nearly 30 seconds when there are two backends, and of nearly 15 seconds when there are four backends, provided that the database size has remained constant.

The second goal is that, for the same requests, if the response sets are increased due to an increase of the database size and the number of backends is increased in proportion to the increase of response set, then the response time per request remain the same. For example, if a request had a response time of 60 seconds when there was one backend with 1000 records in the response set, then the same request would have a response time of close to 60 seconds when there are two backends and 2000 records in the response set.

The underlying concept in each goal is that MBDS in the OAE would supply a database system that would grow as the OAS grows, and would either maintain a constant response time per request by 'growing' its backends or halve a given response time per request by 'doubling' its backends. On the basis of our preliminary analysis, the operational MBDS can indeed meet the two goals. The analysis is also published and documented in [5, 6, 14].

4. THE INTEGRATION OF MBDS INTO THE OAE

In this section, we first focus on the ways to integrate MBDS into the OAE. In this focus, we consider five possible configurations as candidates for integrating MBDS into the OAE. Our second focus examines the relative advantages/disadvantages of the integration configurations. In the examination, we are interested in presenting an intuitive comparison of the five configurations.

4.1. Five Approaches to the Integration of a Database Server

Recall that the basic OAS, consists of a group of workstations, connected by a local-area network (LAN) such as an Ethernet [12]. Such a design has been shown in Figure 1. Given this basic characterization, we now consider the integration of MBDS into the OAS. In the first approach, MBDS is added on as a separate group of workstations in the OAS, with its own LAN. We characterize this approach as the *non-integrated dual-LAN design*. In this approach, the additional workstations are dedicated to the database operations. As such, they are inaccessible for non-database activities. We provide the interface process (which may include one or more language interfaces) as a part of the user-accessible workstations, i.e., hosts. The resulting OAS is shown in Figure 3. In this and the remaining four approaches, the placement of the interface software (i.e., the number of hosts and which hosts have the interface software) is left to the discretion of the database administrator.

The second approach is the *non-integrated single-LAN design*. In this approach, as shown in Figure 4, MBDS and the OAS share a common LAN. However, the MBDS controller and backends still remain as separate workstations in the OAE.

The third approach, the *partially-integrated design*, integrates and replicates the backend software as permanent background processes into some of the OAS workstations. The remainder of the MBDS may have

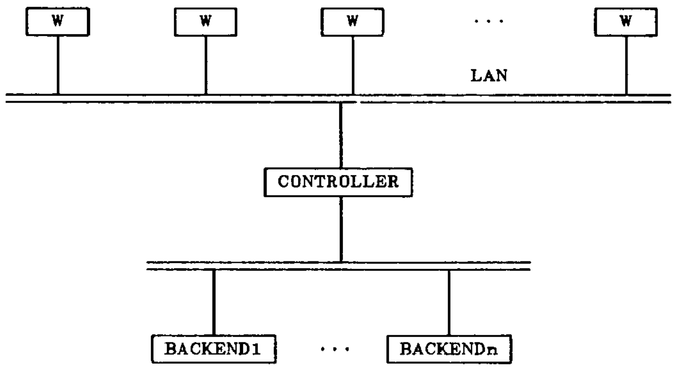


Figure 3. The Non-Integrated Dual-LAN Design

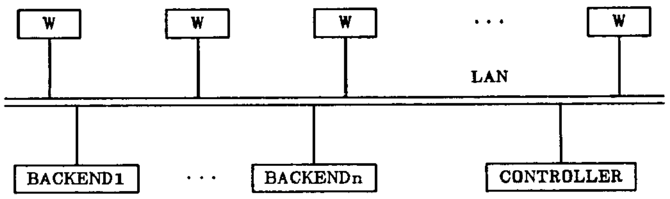


Figure 4. The Non-Integrated Single-LAN Design

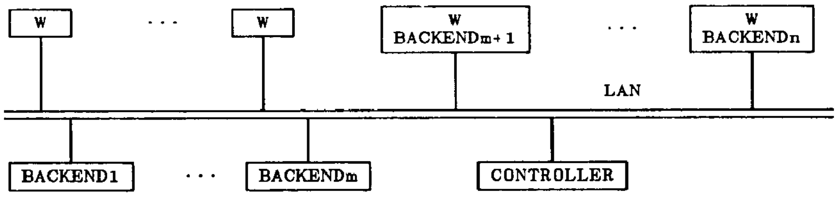


Figure 5. The Partially-Integrated Design

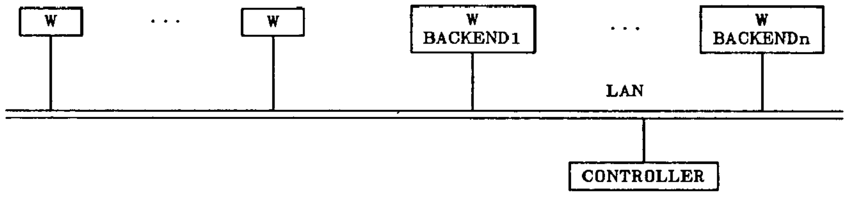


Figure 6. The Isolated-Controller Design

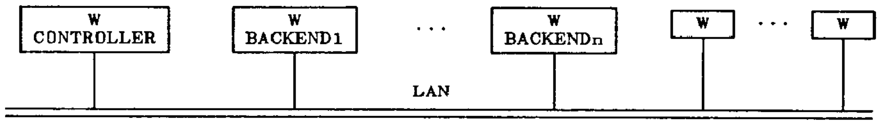


Figure 7. The Fully-Integrated Design

the backend software replicated in user-inaccessible workstations. The distribution of the backend software within the user workstations is controlled by the database administrator in the OAE. The controller is the key component in MBDS, and is devoted to overseeing the management of the database system. Therefore, the controller software is placed in a separate workstation, that is not directly utilized in the OAS. The partially-integrated design is shown in Figure 5.

In the fourth approach, the *isolated-controller design*, the MBDS backend software is integrated and replicated into the existing workstations. As in the partially-integrated design, the controller software is implemented in a user-inaccessible workstation. The backend software is installed as permanent background processes in one or more workstations. The isolated-controller design is shown in Figure 6.

In the fifth approach, the *fully-integrated design*, the MBDS software is completely integrated into the OAS. The controller software is installed as permanent background processes on one workstation. The backend software is installed and replicated as permanent background processes on one or more workstations. The fully-integrated design is given in Figure 7.

In the non-integrated dual-LAN design, we are using the LAN as a logical two-way communications device for MBDS. Messages are passed from the interface process of a particular workstation to the controller and from the controller back to the interface process. In the remaining four designs, we are using the LAN as a logical five-way communications device. Messages are passed from the interface process to the controller, from the controller to the backends, between the backends, from the backends to the controller, and from the controller back to the interface process.

4.2. A Comparison of the Five Designs

Informally, let us consider the trade-offs resulting from one approach to another, which depend on various performance and cost considerations. The non-integrated approaches differ only by the cost of a LAN, but the corresponding performance gains of the dual-LAN approach probably outweighs the cost of the extra LAN. In particular, the load on the LAN for the OAS is significantly lower in the dual-LAN design. However, in both these approaches, a high price is paid as the database and transactions of MBDS grow in size and intensity. The integration of more backends into MBDS and therefore the OAS is costly, since the new workstations are only accessible to the database management system.

In such a situation, either the partially-integrated design or the isolated-controller design becomes an attractive alternative. In either case, keeping the controller on a non-accessible workstation is a good choice for performance. In the partially-integrated design, as the database grows, more user workstations can be configured into the database system. Further, if a workstations is entirely devoted to either the backend software or the OAS software but not both, additional workstations can be easily added to either MBDS or OAS. Thus, in the partially-integrated design, these workstations can be added as either dedicated database backends or user workstations. Again, in either case, the addition of more backends into MBDS is more cost-effective, even if the backend software is added as part of the user workstations. We feel that the fully-integrated design is the least desirable. The controller as part of a user-accessible workstation would substantially degrade the performance of MBDS as the non-database use of the workstation at which the controller resides increases.

Overall, the non-integrated dual-LAN design may yield the highest performance and have the greatest cost. (See Figure 3 again.) The performance of the non-integrated single-LAN and partially-integrated designs are about the same. However, the partially-integrated design is more versatile in performance gain and is still cost-effective. The isolated-controller is not versatile in performance, although it excels in cost-effectiveness. Finally, while the fully-integrated design is cost-effective, its performance may leave a lot to be desired.

5. THE ANALYSES OF THE FIVE INTEGRATION DESIGNS

In this section we provide an examination and analysis of the five configuration designs presented in Section 4.1. Our basic intent is to provide a framework by which the configurations presented earlier can be analyzed and compared. In the process, we can determine just how well our intuitive comparison in Section 4.2 measures up to a queueing model analysis of the configurations.

In order to compare the five approaches to integrating MBDS into an office automation environment, we use simple queueing models for single and multiple devices. To avoid the possible confusion in the use of the term 'server', we refer to the servers of queues as *devices*. An analysis of the message types in the five approaches indicates that both the single- and the multiple-device models must be applied to each approach, due to the variety of workstations in each configuration, (i.e., we have several workstations, several backends and one controller in the configurations). Whether the controller is dedicated or incorporated into a workstation is taken into account in the analysis.

As mentioned in Section 4.1, the local-area networks are used either as 2-way logical communication devices as in the dual-LAN design or as a 5-way logical communication device in the other four single-LAN designs. To reduce the complexity of the queueing analysis, we assume that the backends do not communicate with each other; a similar assumption holds for the workstations. These assumptions are reasonable. In other words, we limit the communication paths in the configurations to: workstation-to-controller, controller-to-backend, backend-to-controller, and controller-to-workstation.

However, this does not necessarily limit the actual message types that are present in each of the five configurations. By examining the operational aspects of the five configurations we note that there are five distinct types of devices; workstation (W), backend (B), controller (C), workstation-backend (WB), and workstation-controller (WC). In the WB device, the workstation functions are combined with the backend functions. (See Figures 5, 6, and 7 again.) In the WC device, the workstation functions are combined with the controller functions. (See Figure 7 again.) Given this characterization, we can summarize the distinct message types for each of the five configurations. These are presented in Table 1.

Let us briefly explain the notation used in Table 1. The entities on the right of the arrows represent the devices which receive and service the message. The entities on the left of the arrows represent the origin of the message. For example, W ====> C, is a message that is serviced by the controller (C) and is received from a workstation (W). Compound letters (e.g., WB, BW, and CW) are used to represent an integrated workstation. The first letter in the abbreviation indicates the part of the integrated unit that is intended as the originator or recipient of a message. In our analysis, we assume that each message type has a distinct message service time associated with it.

5.1. The Queueing Equations

In order to apply simple queueing theory to the problem at hand, we must continue to add to our list of assumptions. In particular, the following assumptions need be made: the *inter-arrival times* of messages follows a Poisson process, the *service times* of messages are exponentially distributed, and the items in the queue are serviced on a first-come-first-serve (FCFS) basis. All of these assumptions are used to simplify the queueing equations that are used to evaluate and compare the five configuration designs. In our queueing analysis, we are interested in calculating the waiting time of individual devices and the total waiting time of all devices. The *waiting time* of individual devices, is the time spent by an item in a queue of a particular device type before the item is serviced. The *total waiting time* of all devices is the sum of the waiting times of an item for **all** of the devices. Now let us proceed with the specification of

Configuration	Input to MBDS			Output to Workstation		
Non-Integrated Dual-LAN	W	====>	C	B	====>	C
	C	====>	B	C	====>	W
Non-Integrated Single-LAN	W	====>	C	B	====>	C
	C	====>	B	C	====>	W
Partially-Integrated	W	====>	C	B	====>	C
	WB	====>	C	BW	====>	C
	C	====>	B	C	====>	W
	C	====>	BW	C	====>	WB
Isolated-Controller	W	====>	C	BW	====>	C
	WB	====>	C	C	====>	W
	C	====>	BW	C	====>	WB
Fully-Integrated	W	====>	CW	BW	====>	CW
	WB	====>	CW	CW	====>	W
	CW	====>	BW	CW	====>	WB

Table 1. A Summary of Message Types

the queueing equations (i. e., the waiting-time equations) for the single- and multiple-device models.

For the single-device model, given the average service time (s) and the average number of messages (n), the average number of items waiting to be served is given by the equation, $w = \frac{p}{1-p}$, where p is the device utilization and is equal to $n \times s$. The waiting time in the queue is given by $t_w = \frac{w}{n}$.

In the multiple-device model, if there are M identical devices, the average number of items waiting to be served is given by $w = B \times \left(\frac{p}{1-p}\right)$ where $p = \frac{n \times s}{M}$, $B = \frac{1-A}{1-pA}$ and $A = \left(\sum_{i=0}^{M-1} \frac{(pM)^i}{i!}\right) / \left(\sum_{i=0}^{M} \frac{(pM)^i}{i!}\right)$. The waiting time is, as before, given by $t_w = \frac{w}{n}$. Depending on the particular message type, we apply either single-device or multiple-device queueing theory and obtain the corresponding equations for the waiting time. In either the single- or multiple-device model, we use the individual waiting times to calculate the total waiting time, T_w, where $T_w = \sum_{\text{all devices}} t_w$.

In order to specify the queueing-time equations (i.e., the total waiting time) for the five different configurations, we must add once again to our list of assumptions. In particular, we assume the following:

(1) There is a maximum of, k, workstations, irrespective of whether a workstation is integrated or not with a backend or controller. Each workstation generates a total of a messages. In other words, the number of workstations and the number of messages generated by a workstation are fixed.

(2) There is a total of, m, backends. If n of these are dedicated backend workstations, then there are $(m-n)$ integrated workstations and $(k-(m-n))$ dedicated workstations. Each backend generates a total of b messages.

(3) There is exactly one controller. If the controller is integrated into a workstation, then the number of dedicated workstations is reduced by one. The controller does not generate any messages of its own and thus merely acts as a relaying mechanism for message types between the workstations and the backends.

(4) Associated with each distinct device type is a unique message processing time. We call this time the *service time*. These service times are represented as subscripted quantities, viz., S_W, S_C, S_{WB}, etc.

(5) Unless otherwise specified, all performance parameters in the equations represent average values. The total waiting-time equations for the five configurations are given in the Appendix.

5.2. A Queueing Analysis of the Five Designs

To obtain a quantitative determination of which of the configurations is better suited to an office automation environment, we compute the total waiting time for each of the five configurations. We begin by assuming that the workstations generate a fixed number of queries and that the backends generate a fixed number of responses. Next, we determine the values for the variables that have been listed in the previous section. Specifically, we must assign values for k, m, n, a, b and the service times. For our analysis, we pick $k = 4$, $m = 2$, $a = 1$, and $b = 1$. The number, n, of dedicated backends will vary, depending on the configuration. For the service times, we choose $S_W = S_B = S_C = S_{WB} = S_{WC} = S = 0.1$.

Given this scenario, we can now present a synopsis of the device structure in each of the five configurations. This synopsis is presented in Table 2. Briefly, let us examine the distribution and functionalities for the devices in each configuration. In both the non-integrated single-LAN and dual-LAN designs, there are a total of seven devices, with 4 dedicated workstations,

Configuration	Device Distribution and Functionality			
Non-Integrated Dual-LAN	4 W	2 B	1 C	
Non-Integrated Single-LAN	4 W	2 B	1 C	
Partially-Integrated	3 W	1 WB	1 B	1 C
Isolated-Controller	2 W	2 WB	1 C	
Fully-Integrated	1 W	2 WB	1 WC	

Table 2. The Device Structure

2 dedicated backends, and 1 dedicated controller. The partially-integrated design has six devices, with 3 dedicated workstations, 1 device having both the workstation and backend functions, 1 dedicated backend, and 1 dedicated controller. In the isolated-controller design, there are five devices, with 2 dedicated workstations, 2 devices having both the workstation and backend functions, and 1 dedicated controller. Finally, in the fully-integrated design, there are four devices, with 1 dedicated workstation, 2 devices having both the workstation and backend functions, and one device having both the workstation and controller functions.

Using this device structure and working with the variable values defined above, we can calculate the total waiting times for each of the five configurations using the equations in the Appendix. The computed results for the total waiting times are summarized in Table 3, where T_w is the total waiting time and T'_w is the total waiting time that is common to all of the configurations.

Configuration	$T_w - T'_w$
Non-Integrated Dual-LAN	0.04
Non-Integrated Single-LAN	0.042
Partially-Integrated	1.58
Isolated-Controller	0.051
Fully-Integrated	0.282

Table 3. The Total Waiting Times

Based on the numerical results presented in Table 3, it can be seen that the dual-LAN configuration has the best performance. This is to be expected since the dual-LAN structure allows for the concurrent processing of messages between the hosts and the controller on the one hand, and between the backends and the controller on the other. Of the remaining four, the non-integrated single-LAN performs the best, barely outperforming the isolated-controller configuration. Surprisingly, the isolated-controller configuration seems to outperform both the fully-integrated and the partially-integrated configurations. A plausible explanation is that the isolated-controller configuration has fewer distinct message types than either the fully-integrated or the partially-integrated configurations. (See Table 2 again.) For similar reasons, the fully-integrated configuration outperforms the partially-integrated configuration. Overall, we observe that the intuitive comparison in Section 4.2 has been generally accurate, with the exception of the partially-integrated design.

6. CONCLUSIONS

We have shown how MBDS can play an important role in the OAE as an database server. Specifically, we have shown how MBDS can provide both traditional and new database support. We have also shown why and how MBDS should be integrated into an OAS, i.e., what MBDS has to offer to an OAE. In particular, MBDS can be integrated into an OAS in a number of ways, depending upon the needs of the office automation environment. Once MBDS is configured in an OAS, the reconfigurability feature, coupled with the replicated backend software, permits the database server to grow as the needs of the office information system grow. Additionally, when MBDS expands, the response time per request for the system decreases proportionately, as long as the database size remains constant. As the database grows in size and consequently the responses, MBDS can maintain the response time for the same type of database services by expanding its backends.

As a multi-lingual database system, MBDS offers the ability to access the database using a variety of language interfaces. From the native data language of MBDS to sophisticated data languages such as SQL, Daplex, CODASYL and DL/I, the user can select a language to query the database. The high degree of software portability exhibited by MBDS, allows the system to be implemented on a wide range of microcomputers, offering database support for a wide range of office automation systems. Overall, we feel that MBDS is close to an ideal database server for the office automation environment.

At the queueing analysis level, we still have some work to do. The queueing analysis presented in Section 4 has been preliminary, and we are currently pursuing a more complete analysis of the five configurations. Among other things, we hope to ascertain how well each configuration performs under different assumptions, i. e., for different values of the queueing variables. We would also like to determine when the performance of a particular configuration begins to degrade, given an increase in the number of devices in the configuration.

Finally, we must also investigate the other relevant considerations when choosing a particular configuration for a specific OAE. Basically, the choice of a configuration depends not only on quantitative data of the configurations' performance, but on external considerations, such as the environment into which the database server is integrated. If an OAS consists of a fixed number of workstations, say two, and there is no growth of the system in terms of additional workstations, then the logical choice is to adopt the fully-integrated approach where the controller is on one workstation, the backend on the other, and the database interfaces on both. If, instead, the OAE is large enough to demand a dedicated database server, then the logical choices are either the non-integrated single-LAN or dual-LAN approach.

REFERENCES

[1] Boyne, R., et al., "A Message-Oriented Implementation of a Multi-Backend Database System (MBDS)," in *Database Machines,* Leilich and Missikoff (eds.), Springer-Verlag, 1983.

[2] Boyne, R., et al., "The Implementation of a Multi-Backend Database System (MBDS): Part III - The Message-Oriented Version with Concurrency Control and Secondary-Memory-Based Directory Management," Technical Report, NPS-52-83-003, Naval Postgraduate School, Monterey, California, March 1983.

[3] Demurjian, S. A., et al., "The Implementation of a Multi-Backend Database System (MBDS): Part IV - The Revised Concurrency Control and Directory Management Processes and the Revised Definitions of Inter-Process and Inter-Computer Messages" Technical Report, NPS-52-84-005, Naval Postgraduate School, Monterey, California, March 1984.

[4] Demurjian, S. A. and Hsiao, D. K., "New Directions in Database-Systems Research and Development," in the *Proceedings of the New Directions in Computing Conference, Trondheim,* Norway, August, 1985; also in Technical Report, NPS-52-85-001, Naval Postgraduate School, Monterey, California, February 1985.

[5] Demurjian, S. A., et al., "Performance Evaluation of a Database System in Multiple Backend Configurations," *Proceedings of the 1985 International Workshop on Database Machines,* March 1985.

[6] Demurjian, S. A. and Hsiao, D. K., "Benchmarking Database Systems in Multiple Backend Configurations," *IEEE Database Engineering Bulletin,* March 1985.

[7] He, X., et al., "The Implementation of a Multi-Backend Database System (MBDS): Part II - The First Prototype MBDS and the Software Engineering Experience," Technical Report, NPS-52-82-008, Naval Postgraduate School, Monterey, California, July 1982; also appeared in *Advanced Database Machine Architecture,* Hsiao (ed.), Prentice Hall, 1983.

[10] Kerr, D.S., et al., "The Implementation of a Multi-Backend Database System (MBDS): Part I - Software Engineering Strategies and Efforts Towards a Prototype MBDS," Technical Report, OSU-CISRC-TR-82-1, The Ohio State University, Columbus, Ohio, January 1982; also appeared in *Advanced Database Machine Architecture,* Hsiao (ed.), Prentice Hall, 1983.

[11] Macy, G., "Design and Analysis of an SQL Interface for a Multi-Backend Database System," Master's Thesis, Naval Postgraduate School, Monterey, California, March 1984.

[12] Metcalfe, R. M., and Boggs, D. R., "Ethernet:Distributed Packet Switching for Local Computer Networks," *Communications of the ACM,* Vol. 19, No. 7, July 1976.

[13] Rollins, R., "Design and Analysis of a Complete Relational Interface for a Multi-Backend Database System," Master's Thesis, Naval Postgraduate School, Monterey, California, June 1984.

[14] Tekampe, R. C., and Watson, R. J., "Internal and External Performance Measurement Methodologies for Database Systems," Master's Thesis, Naval Postgraduate School, Monterey, California, June 1984.

[15] Weishar, D., "Design and Analysis of a Complete Hierarchical Interface for a Multi-Backend Database System," Master's Thesis, Naval Postgraduate School, Monterey, California, June 1984.

[8] Hsiao, D. K. and Menon, M.J., "Design and Analysis of a Multi-Backend Database System for Performance Improvement, Functionality Expansion and Capacity Growth (Part I)," Technical Report, OSU-CISRC-TR-81-7. The Ohio State University, Columbus, Ohio, July 1981.

[9] Hsiao, D. K. and Menon, M.J., "Design and Analysis of a Multi-Backend Database System for performance Improvement, Functionality Expansion and Capacity Growth (Part II)," Technical Report, OSU-CISRC-TR-81-8. The Ohio State University, Columbus, Ohio, August 1981.

APPENDIX. THE TOTAL WAITING TIME EQUATIONS

In this appendix we provide a synopsis of the total waiting time equations used for each of the five configurations that have been proposed in Section 4. These equations were used to calculate the data presented in Table 3. In the equations below, k is the number of workstations, dedicated or not, m is the number of backends, dedicated or not, n is the number of backends, and S is the service time. The total waiting time is $T_w = \sum_{\textit{all devices}} t_w$.

The total waiting time for the non-integrated dual-LAN configuration is:

$$\max\left(kS^2/(1 - kS),\ mS^2/(1 - mS) \right) + \max\left(B_1/(m - kS),\ B_2/(k - mS) \right)$$

$$\text{where } B_1 = \frac{(kS)^m / m!}{(kS)^m / m! + (1 - \frac{kS}{m})\sum_{i=0}^{m-1} \frac{(kS)^i}{i!}}$$

and $B_2 = \dfrac{(mS)^k / k!}{(mS)^k / k! + (1 - \frac{mS}{k}) \sum_{i=0}^{k-1} \frac{(mS)^i}{i!}}$

The total waiting time for the non-integrated single-LAN configuration is:

$$kS^2/(1 - kS) + mS^2/(1 - mS) + B_1/(m - kS) + B_2/(k - mS)$$

where B_1 and B_2 are as above.

The total waiting time for the partially-integrated configuration is:

$$kS^2/(1 - kS) + mS^2/(1 - mS) + B_1/(n - kS) +$$

$$B_2/(m - n - kS) + B_3/(k - m + n - mS) + B_4/(m - n - mS)$$

where $B_1 = \dfrac{(kS)^n / n!}{(kS)^n / n! + (1 - \frac{kS}{n}) \sum_{i=0}^{n-1} \frac{(kS)^i}{i!}}$

and $B_2 = \dfrac{(kS)^{m-n} / (m-n)!}{(kS)^{m-n} / (m-n)! + (1 - \frac{kS}{m-n}) \sum_{i=0}^{m-n-1} \frac{(kS)^i}{i!}}$

and $B_3 = \dfrac{(mS)^{k-m+n} / (k-m+n)!}{(mS)^{k-m+n} / (k-m+n)! + (1 - \frac{mS}{k-m+n}) \sum_{i=0}^{k-m+n-1} \frac{(mS)^i}{i!}}$

and $B_4 = \dfrac{(mS)^{m-n} / (m-n)!}{(mS)^{m-n} / (m-n)! + (1 - \frac{mS}{m-n}) \sum_{i=0}^{m-n-1} \frac{(mS)^i}{i!}}$

The total waiting time for the isolated-controller configuration is:

$$kS^2/(1 - kS) + mS^2/(1 - mS) + B_1/(m - kS) +$$

$$B_2/(k - m - mS) + B_3/(m - mS)$$

where $B_1 = \dfrac{(kS)^m / m!}{(kS)^m / m! + (1 - \frac{kS}{m}) \sum_{i=0}^{m-1} \frac{(kS)^i}{i!}}$

and $B_2 = \dfrac{(mS)^{k-m} / (k-m)!}{(mS)^{k-m} / (k-m)! + (1 - \frac{mS}{k-m}) \sum_{i=0}^{k-m-1} \frac{(mS)^i}{i!}}$

and $B_3 = \dfrac{(mS)^m / m!}{(mS)^m / m! + (1 - S) \sum_{i=0}^{m-1} \frac{(mS)^i}{i!}}$

The total waiting time for the fully-integrated configuration is:

$$kS^2/(1 - kS) + mS^2/(1 - mS) + B_1/(m - kS) +$$

$$B_2/(m - mS) + B_3/(k - m - 1 - mS)$$

where $B_1 = \dfrac{(kS)^m / m!}{(kS)^m / m! + (1 - \frac{kS}{m}) \sum_{i=0}^{m-1} \frac{(kS)^i}{i!}}$

and $B_2 = \dfrac{(mS)^m / m!}{(mS)^m / m! + (1 - S) \sum_{i=0}^{m-1} \frac{(mS)^i}{i!}}$

and $B_3 = \dfrac{(mS)^{k-n-1} / (k-n-1)!}{(mS)^{k-n-1} / (k-n-1)! + (1 - \frac{mS}{k-n-1}) \sum_{i=0}^{k-n-2} \frac{(mS)^i}{i!}}$