Computing and Informatics, Vol. 23, 2004, 255–285

DSTP-AN: A DISTRIBUTED SYSTEM FOR TRANSACTION PROCESSING BASED ON DATA RESOURCE MIGRATION IN ATM NETWORKS

Prem Chandra Saxena

School of Computer and System Sciences
Jawaharlal Nehru University
New Delhi 110067, India

D. Roy Choudhury, Goldie Gabrani

Department of Computer Engineering
Delhi College of Engineering
Bawana Road, New Delhi 110042, India
e-mail: ggabrani@yahoo.co.in

Manuscript received 27 August 2001; revised 17 March 2004
Communicated by Jiří Šafařík

Abstract. The dynamic migration of data resources has become a strong tool for transaction processing in broadband networks such as ATM. In this paper, a distributed system that takes advantage of data resource migration for transaction processing in ATM networks has been proposed. The proposed system provides mechanisms to select the transaction processing method, to migrate data resources in a way that reduces the time delay and message traffic in locating and accessing them. The first mechanism selects one of the two transaction processing methods: the traditional method that uses two phase commit protocol and other new method based on data resource migration. The second mechanism attempts to improve performance by making each site follow a local policy for directing requests to locate and access data resources as well as migrating them through the system. For this, a new scheme that focuses on reducing the time delay and message traffic needed to access the migratory data resources is proposed. The performance of the proposed scheme has also been evaluated and compared with one of the existing schemes by

a simulation study under different system parameters such as frequency of access to the data resources, frequency of data resource migrations, scale of network, etc.

Keywords: ATM networks, data resource, migration, transaction processing

1 INTRODUCTION

Very high speed data transmission in the order of Gb/s and the bit error rate from 10^{-6} to less than 10^{-9} has become possible with the developments in fiber optic and high speed switching technologies, such as ATM [1, 13]. Such broadband networks permit us to transmit huge volumes of data in a very short period of time and this has created a paradigm shift with distributed system designers who now focus on optimal bandwidth utilization as the means to enhance system performance. Data intensive multimedia applications, such as interactive video on demand and real time video conferencing [1, 24] have already exploited the high bandwidth available in broadband networks. A number of application systems [4, 9, 15, 22, 26] that utilize the very high speed data transmission of such networks have also been developed. High bandwidth also provides a flexible and high quality mobile computing environment [1, 5, 6, 23]. The impact of high bandwidth can also be seen on the technologies of distributed processing as discussed below.

Some research has been done in the area of distributed computing while utilizing migration of data items. The use of file migration in distributed systems has been demonstrated in [10, 11, 17]. It has been shown that data files in distributed systems can be replicated and allocated to various sites, using file migration, so as to reduce communication costs and to increase reliability. A detailed survey of various file migration policies has been elaborated in [10]. Several query processing algorithms also make use of data communication between sites to reduce the query processing response time [2, 3, 7, 25, 32]. The other usage of data migration technique is for load balancing among distributed file servers by relocating the distributed data [18, 19, 21, 30, 31]. However, since these studies do not assume the use of broadband networks, they estimate data relocation as a great overhead for distributed systems, and therefore data migration is performed in a limited controlled manner and is not executed frequently.

On the other hand, with the availability of high bandwidth in broadband networks the situation has changed. In conventional distributed environments (with narrowband networks), the primary factor for performance improvement was the minimization of the amount of data to be transmitted. However, in recent environments, more efficient use of broadband networks is now one of the most important issues for performance improvement. This is mainly because, in a broadband network, the propagation delay is almost equal to that of a conventional network and the transmission delay is very small even for data of huge volume. Therefore, it is rather efficient to collectively transmit data of large volume than to transmit small

data multiple number of times [14]. As an illustration, for example, propagation delay across a country, say, United States at the speed of light is about 20 ms [4]. In broadband networks, at say 1 Gb/s, it will only take 1 ms to transmit a 1 megabit file, resulting in a total delay of 21 ms. For a traditional network, like the internet, operating at 50 Kb/s, the transmission delay is 20 s, giving a total delay of 20.020 ms. Therefore, in conventional narrowband networks, most of the existing algorithms have focused on minimizing the volume of data to be transmitted. On the other hand, in broadband networks, as it has become possible to transmit a great volume of data in a very short period of time, migration of data is expected to be one of the most useful mechanisms.

Some of the studies premised above include development of query processing strategies for high speed local area networks [33, 34]. The research in [4] and [12] has reported how to utilize advantages of broadband networks for transaction processing. In [4], data items requested by the transaction are forwarded to the transaction initiation site. In [12], an architecture in which database contents are broadcasted periodically through a fiber optic ring has been proposed. Another example is Data Resource (DR) migration [16], which means the migration of whole DR (such as database or file) from one site to another through networks. The dynamic relocation of DRs using DR migration is an emerging area of research that has been prompted by broadband networks. The use of this technique has been demonstrated in [14, 15, 24, 27, 28, 29] and it has been shown that this technique can be used in the area of distributed processing in several ways, especially for transaction processing.

In conventional networks, DRs are fixed at their home sites and are accessed by other sites by sending several query (operation) messages. Therefore, a transaction requires many message transmissions for exchanging operation messages and control messages. After the message exchange, the operation is consistently validated by using the Two Phase Commit (2PC) protocol. This method is referred to as fixed processing method. Whereas in broadband networks, since the transmission delay is considerably lower as compared to conventional narrowband networks, the other method called DR migration method can be employed. Using this method, the whole DR can be migrated from one site to another in a very small amount of time. In broadband networks, therefore, the transaction initiation site can collect all the necessary DRs through DR migration method and then execute the transaction locally. It is seen that in fixed processing method, processing a transaction often requires many message transmissions, whereas by using DR migration method, there is no need for exchanging messages after the transaction initiation site has gathered all the required DRs. It has been shown in [15, 16] that in some practical situations, DR migration method can be used to shorten the transaction processing time considerably. Such results demonstrate the importance of drastic change in conventional transaction processing techniques for efficient utilization of the broadband channel.

This is also illustrated by the following example [14]. Suppose a 1.2×10^7 meter long fiber optic cable with a bandwidth of one Gb/s connects two sites i and j. Since speed of light in the fiber is 2.1×10^8 m/s, the propagation delay between the two sites is at least 0.057 s even after processing delays at sites and switching

delays at intermediate ATM switches are neglected. Now a transaction at site i intends to access a DR at site j which is 100 megabytes in size. Transfer of whole DR from site j to site i would require about 0.8 s, neglecting processing delays at sites and switching delays at intermediate switches. Therefore, if the transaction at site i needs to query the DR at site j in more than seven rounds, then clearly it is better to transfer the DR from site j to site i in order to increase performance. The threshold value becomes even smaller, if the processing delays and the routing delays are not neglected. This substantiates the need to migrate DRs in order to increase performance and utilize available bandwidth efficiently. Further, as the upper limit of data propagation speed is the speed of light, it is difficult to significantly reduce the propagation delay. Hence, DR migration will be more effective as the scale of networks becomes larger. Therefore, to take full advantage of DR migration, it is desirable that the DRs have low location dependency, are not extremely large, say in the order of gigabytes, and the sites are distributed over a wide area. Further, it has been shown in [HHTN98b] that use of DR migration method highly contributes to the performance improvement in transaction processing in ATM networks and can be thus considered as one of the primitive DR operations.

It has been shown with the preceding example that in broadband networks, migration of DRs is, most of the times, better than accessing them remotely. In order to support DR migration, a distributed system architecture that is different from those that have been developed for narrowband networks, is required. One such architecture, DB-MAN (A Distributed Database System based on Database Migration in ATM Networks) [15] has been proposed for the purpose. DB-MAN proposed for ATM environments, focuses mainly on selection of the transaction processing method (fixed processing or DR migration), referred to as method selection by DB-MAN, as means of performance improvement. Both the modes of method selection, viz. simple mode and log statistics mode proposed by DB-MAN, rely on broadcast of request messages, by having a centralized controller in the form of a multicast server. The centralized approach has certain inherent disadvantages. The systems with centralized server are neither scalable nor fault tolerant. If the centralized server breaks down the whole system goes down. Though this problem can be solved using backup systems it involves considerable overhead for ensuring consistency, failure detection and recovery. Also, as the number of requests in the system increases, the increase in load at the centralized server negatively affects the performance of the system. Further, as the number of sites in the system increases, use of a centralized multicast server with connections with all sites in the system is not a viable choice [14].

In order to remove the disadvantages of a centralized system, a decentralized system, DSTP-AN (*A Distributed System for Transaction Processing based on Data Resource Migration in ATM Networks*), has been proposed in this paper. DSTP-AN is designed for a distributed environment that constitutes a number of DRs, distributed over a set of sites that are interconnected by an ATM network. It supports both fixed processing and DR migration methods for processing the transaction and attempts to improve performance by providing a strategy to select the more

efficient transaction processing method between the fixed processing and DR migration method. It focuses on reducing the time delay (time needed by the transaction initiation site to locate and access the DR) in processing the transaction requests by providing mechanisms for propagation of requests and migration of DRs through the system. It also provides a mechanism for the concurrency control that supports DR migration.

The rest of the paper is organized as follows. The basic system model is presented in Section 2. In Section 3, the architecture of the proposed system is discussed. Section 4 presents the mechanism of selecting the transaction processing method. In Section 5, the details of the component of the proposed system responsible for controlling request traffic and DR migration are given. Section 6 describes the simulation environment and evaluates the proposed system. Finally some concluding remarks are given in Section 7.

2 SYSTEM MODEL

The distributed system under consideration has many homogeneous sites and multiple DRs with low location dependency, i.e. DRs can move without restrictions between the sites involved in the system. The sites are connected via an ATM network that may be an intra-network or a public network and communicate with each other by sending messages. ATM networks are characterized by their switch-based network architecture. All sites are connected to a switch and communication between each pair of sites is essentially established through the switch. In ATM, a connection (a virtual channel in ATM terminology) to a destination site is established in advance of data transmission. Such connections can be of two types: i) Permanent Virtual Connection (PVC), which is established permanently for a certain group of ATM switches; and ii) Switched Virtual Connection (SVC), which is established dynamically according to a request generated by a site and is released if congestion occurs in the network or when the connection is not used during a certain predefined period of time. PVC is a connection set up by some external mechanism, typically network management, and always requires some manual configuration, whereas a SVC does not require the manual interaction as needed to set up a PVC and as such is much more widely used.

Each site in the system has a local DR management system called DSTP-AN. The system structure of DSTP-AN is similar to the DB-MAN, but in contrast to the DB-MAN, DSTP-AN does not assume any centralized controller in the form of a multicast server for the ATM network. DSTP-AN supports both fixed processing and DR migration methods for transaction processing. It is assumed that the transaction initiation site uses a heuristic technique (explained in Section 4) to determine the method by which the transaction is to be processed. After determining the method, the transaction initiation site generates request messages for the DRs, it wants to access. The transaction initiation site then executes the transaction either by fixed processing method, where it sends operation messages to the target

sites (the sites holding the DRs) and accesses the DRs remotely or by DR migration method, where operations on the DRs are performed locally, after the DRs have been migrated to it. In the proposed system, each site has equal priority to generate a request for a DR situated at any site. Thus, DSTP-AN is truly distributed as defined by Enslow [8].

DSTP-AN associates each DR with a particular site called the default site for that DR. The default site is the site where the DR resides if no other site is requesting for the migration of the DR. Whenever a site initiates a transaction on a DR, it sends a request message to the default site for that DR and when all the transactions requiring the migration of the DR are executed, the DR returns to this site. The default site for a DR may be chosen according to some parameters. For example, the site that uses a particular DR more frequently or a site that uses a particular DR continuously can be the default site for that DR. Further, each site has a DR information table that records the size of each DR and the default site for each DR along with its geographical distance from the current site. In the next section, the description of the architecture of DSTP-AN proposed in this paper is presented.

3 ARCHITECTURE OF DSTP-AN

In this section, the architecture of DSTP-AN is presented. DSTP-AN uses both fixed processing and DR migration methods for transaction processing and thus provides some mechanisms that are not required in conventional distributed systems (they utilize only fixed processing method for transaction processing). As illustrated in Figure 1, DSTP-AN mainly consists of the Transaction Manager, the Resource Migration Manager, the Controller and the Resource Interface Manager. Each of these components is discussed below.

3.1 Transaction Manager

The Transaction Manager inputs transactions (or queries) from the User Interface. It parses, optimizes and if required, it breaks the incoming queries into sub-queries. It then adaptively determines the transaction processing method (fixed processing or DR migration) to be used for each transaction by employing a heuristic mechanism, which is described in Section 4. Subsequently, the Transaction Manager forwards the queries/sub-queries to the Controller and a list of target DRs (the DRs required for the transaction) to both the Resource Migration Manager and the Controller.

3.2 Resource Migration Manager

The main task of the Resource Migration Manager is to make available to the transaction initiation site the DRs that are required for the execution of the transaction but are presently not residing with it. In case the transaction is to be processed by fixed processing method, the Resource Migration Manager determines the sites

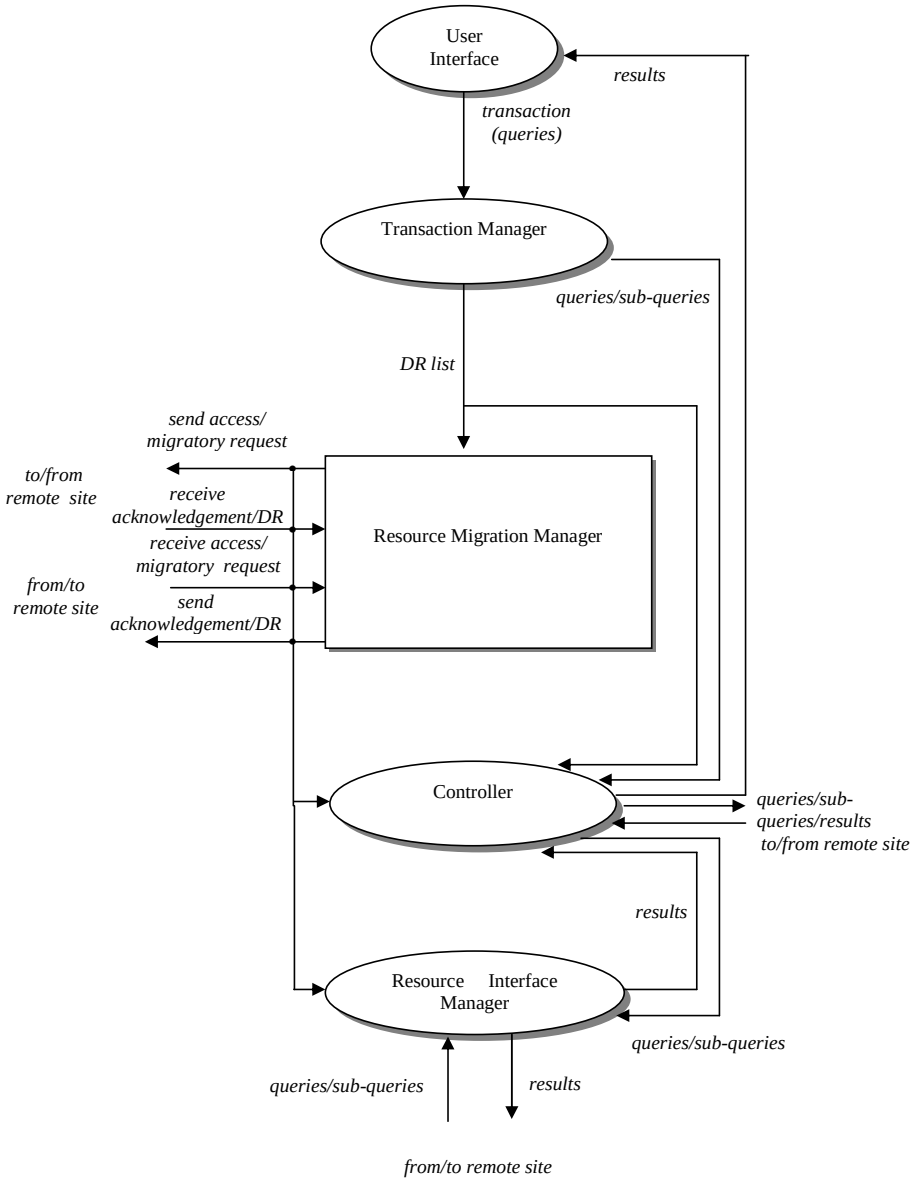


Fig. 1. Structure of DSTP-AN

of the target DRs and passes the information to the Controller. However, if the transaction is to be processed by DR migration method, it collects the requested DRs that are not present on the transaction initiation site. In order to perform the above functions, the Resource Migration Manager of the transaction initiation site generates one of the following two types of request messages:

1. Access request message: This request message is generated when the transaction is to be processed by fixed processing method. When the Resource Migration Manager of the DR holding site receives the access request message, it generates an acknowledgement message to the transaction initiation site, allowing it to access the DR. The information regarding the access request message and the acknowledgement message is also forwarded to the Controller.
2. Migratory request message: This request message is generated to migrate the DR from the DR holding site to the transaction initiation site, when the transaction is to be processed by DR migration method. Whenever the Resource Migration Manager of the DR holding site receives a migratory request message, it sends the DR to the transaction initiation site through the Resource Interface Manager.

As the system under consideration is a distributed system, the Resource Migration Manager of the current site would also send acknowledgement messages/DRs in response to the access/migratory requests received from the Resource Migration Managers of other sites. The manner in which the Resource Migration Manager processes the access and migratory requests is discussed in detail in Section 5. It may also be mentioned here that the Resource Migration Manager also updates the information of the current locations and the sizes of the DRs in the DR information table of its own site.

3.3 Controller

The role of the controller is to perform queries/sub-queries either locally (DR migration method) or remotely (fixed processing method) by requesting the Resource Interface Manager of the transaction initiation site (current site) or of the remote site, in a manner in which the consistency of the DRs is not violated. Thereafter the results are sent to the user. The functions of the Controller also include the concurrency control and coordination among other sites. It resolves the conflict between two or more sites seeking to acquire the same DR (through DR migration). The Controller maintains a list of all the DRs that are present on the current site. It modifies the list as and when the current site either receives or migrates a DR. This is to ensure that the DRs collected by the transaction initiation site are not migrated to other sites while the transaction initiation site is collecting other DRs needed by the transaction. It also keeps a record of all the sites to which the Resource Migration Manager of the current site has sent acknowledgement messages. This is to ensure that the DR for which an acknowledgement message has been given is not migrated to another site till all operations on it by fixed processing are over.

In order to support the above strategies, one of the mechanisms proposed in [16] is to extend the conventional two phase locking protocol [35, 36] by having one more lock, the location lock on the DR, in addition to the conventional read and write locks. Whenever a transaction initiation site receives a DR it needs for the transaction, it sets a location lock on it. The location lock on the DR is not unset until the transaction initiation site gathers all the DRs and processes its transaction. After the transaction is over, the location lock from the DR is removed (implying that the DR can now be migrated to other sites). Similarly, whenever a site sends an acknowledgement message for the DR it holds, it sets a location lock on it. The location lock is released only after the operations on the DR are over.

3.4 Resource Interface Manager

The Resource Interface Manager handles the interface with physical storage for DR management. It manages the DR related functions such as read, write and migrate. It applies the queries/sub-queries received from the Controller of the current site or a remote site to the DRs (present on the current site) and sends the results back to the Controller of the current site or the remote site, respectively. It migrates/receives the DR when it receives a send/receive DR message from the Resource Migration Manager.

4 STRATEGY FOR THE SELECTION OF THE TRANSACTION PROCESSING METHOD

When the User Interface submits a transaction, in the form of queries, to the Transaction Manager, the latter decides to process the transaction by using either fixed processing method or DR migration method. Transaction Manager uses a heuristic technique to arrive at the decision, for which it calculates, heuristically, the approximate time delay needed for processing the transaction using both the methods. It assumes that each query contained in the transaction is associated with a single DR only and transaction processing delay is negligible. Sections 4.1 and 4.2 give description for the calculation of transaction processing time using either fixed processing or DR migration method.

4.1 Fixed Processing Method

The first method, fixed processing, involves remote access of the DR using operation request and reply messages [15]. The communication model of the fixed processing method, illustrated in Figure 2 a, follows the conventional 2PC protocol for consistently validating the transaction. It is assumed that a transaction requires n operations to be performed. Assuming that each operation request generates a single reply message, $2n$ operation request/reply messages will then be exchanged for the transaction. Before the exchange of operation request/reply messages, the

transaction initiation site must establish SVC connections to each site that holds the relevant DRs needed for the execution of the transaction. (In ATM networks, a connection is established before data transfer.) Let k be the number of sites that hold j number of required DRs and let $C(k)$ denote the time required for establishing SVC connections from the transaction initiation site to the k sites that are involved in the transaction. If in a system fixed processing method is used in combination with DR migration method, then the transaction initiation site must know the location of each DR before establishing the connections. Let $LT(j)$ be the time taken by the transaction initiation site to locate j DRs. Following the connection establishment, $2n$ messages are then exchanged for executing the transaction. This takes time $2nt$, where t is the average propagation delay between two arbitrary sites. The transmission delay of the messages is negligible because the size of each message is very small in comparison to the bandwidth of the assumed communication network. Finally, the transaction is validated by 2PC protocol that takes two rounds of message transmissions which takes time $4t$. (The time taken by 2PC would not be considered for read only transactions.) In actual practice, effective propagation delay in the 2PC is more than t on the average as it is bound by the largest delay between the transaction initiation site and each site involved in the transaction. For simplicity of analysis it is assumed to be t. Thus, the transaction processing time by using fixed processing method, denoted by T_{fix} is given by the following equation:

$$T_{\text{fix}} = (2n + 4)t + LT(j) + C(k) \tag{1}$$

($LT(j)$ would be zero if fixed processing method is used in a system that does not support DR migration).

4.2 DR Migration Method

The second method employed for transaction processing is DR migration that involves migration of the required DRs (referred to as target DRs) to the transaction initiation site (Figure 2 b). To implement this method, it is necessary that (i) DRs for the transaction are identified before the initiation of the transaction (requested DRs), (ii) location of requested DRs is known to transaction initiation site and (iii) size of requested DRs is known to transaction initiation site. In order to execute the transaction, the transaction initiation site must first determine the location of the DRs that are needed for the transaction and are not present on the transaction initiation site. Then, it establishes SVC connections with each site that holds the relevant DRs. As in the case of fixed processing method, this process takes $LT(j) + C(k)$ amount of time. After the connections have been established, the transaction initiation site sends request messages for the migration of DRs that takes time t. Note that as it is difficult to formulate the exact propagation delay from the transaction initiation site to each site where the relevant DRs reside and also its exact value is larger than t for the same reason as in the case of 2PC; therefore, for simplicity of analysis, the value of propagation delay is assumed to be t.

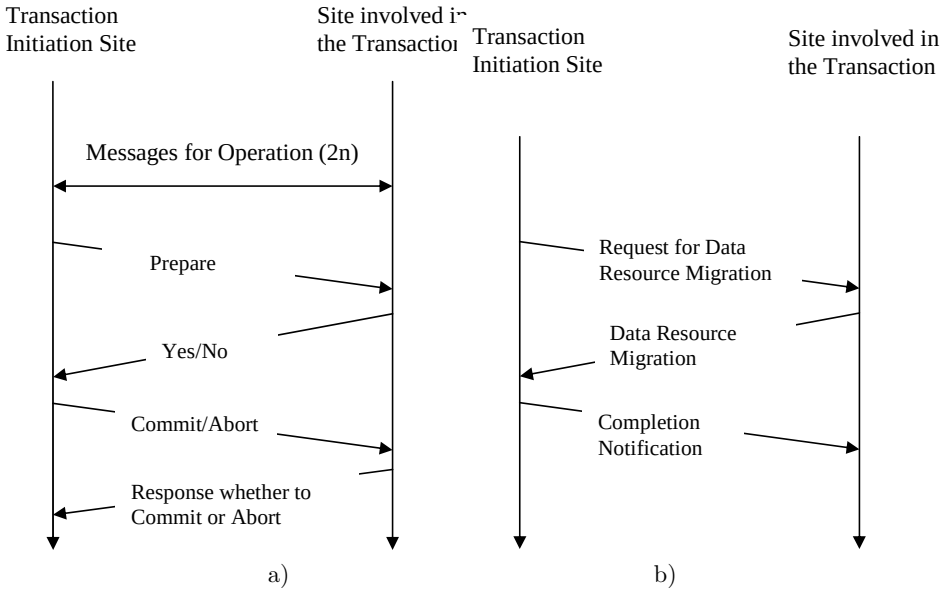


Fig. 2. a) Communication process using fixed processing method; b) Communication process using data resource migration method

DRs are then migrated to the transaction initiation site through the SVC connections that have already been established. Let size(x) denote the size of DR x and is maintained as part of DR information table by each site, D_j be the set of j DRs that do not reside at the transaction initiation site but are needed by the transaction, and B the bandwidth reserved for DR migration. The time required for migration of the DRs is then given by: $\sum_{x \in D_j} \text{size}(x)/B + t$; where $\sum_{x \in D_j} \text{size}(x)/B$ is the transmission delay of the DRs. It should be noted that the DR would encounter the same propagation delay, t as the migratory request (t is the propagation delay between transaction initiation site and the DR holding site). Finally, a completion notification message is sent that takes time t. Therefore, the transaction processing time by using DR migration method, denoted by T_{mig} is expressed by the following equation:

$$T_{\text{mig}} = 3t + \sum_{x \in D_j} \text{size}(x)/B + LT(j) + C(k). \tag{2}$$

In DSTP-AN, the Transaction Manager adaptively selects the transaction processing method that results in minimal delay. For this it calculates $T_d = T_{\text{fix}} - T_{\text{mig}}$. It is assumed that the time $LT(j) + C(k)$ is the same for both T_{fix} and T_{mig}. Therefore, calculation of T_d mainly depends upon the value of propagation delay, t, between the transaction initiation site and the sites holding the target DRs. Since the Transaction Manager cannot know in advance the exact locations of the target DRs, it uses the following heuristic to calculate the delay:

For each of the target DR, the propagation delay between the transaction initiation site and the default site for the corresponding DR is taken as the value of t for the calculation of T_d. (Note that the value of t between any two arbitrary sites is calculated by dividing the geographical distance between them by the speed of light in the fiber.) The Transaction Manager would use DR migration method only if T_d is positive.

5 RESOURCE MIGRATION MANAGER

The Resource Migration Manager (RMM) defines the strategies for message (both request and acknowledgement) handling as well as for DR migration. In this paper, a scheme that may be followed by RMM, for processing access and migratory requests, is proposed. It may be mentioned here that some such schemes have been proposed in [14], where Hara et. al. have suggested four distributed methods for handling the request messages and migration of DRs in broadband networks. Two out of these four, CF (Chain Forwarding) and CQ (Chain Query) have been adopted from mobile computing [37]. The other two, ECF (Extended Chain Forwarding) and ECQ (Extended Chain Query) are improved versions of CF and CQ respectively. It has been shown in [14] that ECF scheme (explained in Appendix) performs the best among the above schemes and hence in this paper the proposed scheme is compared with ECF. The scheme proposed in this paper is now explained in the following section.

5.1 The Proposed Scheme

This scheme is elegantly simple, in which the default site performs all the operations related to its DR. The knowledge about the default sites of all DRs is made known to all sites during initialization of the network or to any site when it enters the network. When a transaction is started, the transaction initiation site first identifies the target DRs and their default sites and then the RMM at the transaction initiation site sends request messages (access or migratory) for all those DRs not residing with it. In this scheme, all access and migratory requests for a DR are addressed to its default site directly. The default site uses waiting queue data structure associated with each DR to deal with the multiplicity of requests. The RMM at the default site on receiving the request message proceeds as follows:

1. if it holds the DR and it receives an access request message from the transaction initiation site, it sends an acknowledgement message to the transaction initiation site,
2. if it holds the DR and it receives a migratory request message from the transaction initiation site, it migrates the DR to the transaction initiation site, and
3. if the target DR is not present at the default site currently, it adds the request to the queue associated with the DR. In this case, the transaction initiation site

has to wait till all the previous requests (in the DR queue) at the default site are fulfilled.

It may be mentioned here that after the transaction initiation site completes its execution on the DR (which was migrated to it by the default site for that DR), the RMM at the transaction initiation site returns the DR to its default site. Since in this scheme, whenever a transaction is initiated the RMM sends requests directly to the default sites, therefore the time required to locate the target sites is nil. In the proposed scheme, RMM at each site i has certain local variables associated with it, which are described below. (Note that for simplicity in explanation, from now onwards site i represents RMM at each site i).

1. **Type of request generated by site i:** The variable request type (req_type) indicates the type of request: migratory or access, generated by site i for a DR. As already mentioned, DR is migrated to site i in case of the former request, while no migration takes place during the latter, instead the operation is performed at the site which holds the DR. A site can generate a migratory request for one DR and access request for the other.
2. **Local state of site i:** Each site i can be in any one of the following states for a DR: i) requesting: site i has generated a request for accessing a DR; ii) holding: site i is holding the DR; and iii) not requesting: none of the above conditions is true. As sites can simultaneously generate multiple requests for different DRs in the system, a site could be in a requesting state for one DR, in a holding state for the other and in a not requesting state for the third.
3. **Location Pointer (loc_ptr) of site i:** Loc_ptr of a site for a DR points to the site where the request must be forwarded in order to access that DR. Therefore, whenever site i intends to access a DR, it sends a request message to the site pointed by loc_ptr for the requested DR and then waits for its request to be fulfilled. Each site i has a loc_ptr for each DR in the system. Hence, each site has m loc_ptrs, one corresponding to each DR. In case site i holds a DR, the value of its loc_ptr for that DR is set to itself.
4. **DR_queues of site i:** Each site i maintains m DR_queues, one for each DR, to queue the incoming requests for a DR. The parameters entered in the queue are site_id and req_type of the sites requesting for the DR. Following cases arise depending upon the value of site_id in the DR_queue (referred to as DR_queue.site_id) for a DR:
    1. DR_queue[i][x].site_id = i; indicates that site i has its own request for DR x.
    2. DR_queue[i][x].site_id = j, $j \neq i$; implies that site i has received a request of site j for DR x.

Whenever site i satisfies its own request or the request of site j for a DR, the entry in DR_queue for the corresponding DR is removed.

In order to record the necessary information the following information structures are constructed at each site i:

1. site state table: At each site i, a site state table is maintained that has m entries corresponding to m DRs. It records the current state of the site with regard to all the DRs in the system. It also stores the req_type of the request generated by site i, if any, for all the DRs in the system (Table 2).

DR_id	state	req_type
1	R	M
2	N	A
⋮	⋮	⋮
x	H	M
⋮	⋮	⋮
m	N	M

Table 1. A pictorial representation of site state table at any site

2. loc_ptr table: Each site i maintains a table to record the values of loc_ptrs of site i for all the m DRs in the system (Table 2). Its value lies in the domain $\{1 - n\}$.

DR_id	loc_ptr
1	2
2	5
⋮	⋮
⋮	⋮
⋮	⋮
m	6

Table 2. A pictorial representation of loc_ptr table at any site

3. DR_queues: As already mentioned, each site i maintains m DR_queues, one for each DR. DR_queue for DR x records the site_id and the req_type of the sites requesting for DR x. The DR_queues are read using FIFO (First-In-First-Out) policy.

5.1.1 Messages

There are three types of messages in the system:

1. **Request message (Req_msg):** These are created and sent by any site i, whenever it intends to access a DR that it does not hold. This message carries site_id of the requesting site, DR_id of the requested DR and the req_type.
2. **DR migrate message (DR_mig_msg):** Whenever a DR is to be migrated, holding site i generates this message.

3. Acknowledgement message (Ack_msg): This message is generated by site i, when it receives an access request message for the DR it holds.

Now detailed description of the proposed scheme follows.

5.1.2 Detailed Description

Before describing the proposed scheme in detail, the notations used for its presentation are discussed. Loc_ptr[i][x] gives the value of loc_ptr of site i for DR x. site_state[i][x].state represents the state of site i for DR x as given by the site state table maintained at site i. Similarly, DR_queue[i][x] represents the DR_queue at site i for DR x. Each DR_queue is read in FIFO order and the entry last read is denoted as DR_queue[i][x].site_id and DR_queue[i][x].req_type, that respectively refers to site_id and req_type entries in the DR_queue. Each Req_msg contains site_id of the requesting site, DR_id of the requested DR and information regarding the type of request. These are referred to as Req_msg.site_id, Req_msg.DR_id and Req_msg.req_type, respectively. In a similar manner, site_id and DR_id contained by Ack_msg are referred to as Ack_msg.site_id and Ack_msg.DR_id, respectively.

Initialization
Initialization process consists of building m default sites corresponding to m DRs by setting up loc_ptrs of all the n sites. Initially, each DR is assumed to be located at a different and a unique site called default site. Default site corresponding to a DR, say DR x, initially holds DR x and the value of its loc_ptr for DR x is set to itself. The remaining sites are in not requesting state for DR x and their loc_ptrs for DR x point towards the default site for DR x. Therefore, initially all the sites, except default sites, are in not requesting state for a DR. The variable req_type, at all the sites, for all DRs, is set to migratory. Moreover, DR_queues at all the default sites corresponding to DRs are initially empty. It may be noted that DR_queue for a DR may store a maximum of n entries.

The initialization procedure is as follows:

```
for(x = 1; x <= m; x++) /* m DRs */
{
  for (i = 1; i <= n; i++) /* n sites */
  {
    if (x==i)
    {
      set site_state[i][x].state = H; /* state of site i for DR x is set to holding */
      set loc_ptr[i][x] = i; /* loc_ptrs of DR holding sites are initialized
      to point to themselves */
      set DR_queue[i][x].site_id = nil;
      set DR_queue[i][x].req_type = nil; /* DR_queue at default sites
      for all DRs are initially empty */
    }
```

```
    else
    {
      set site_state[i][x].state = N; /* not requesting */
      set loc_ptr[i][x] = x; /* loc_ptrs of non DR holding sites are initialized */
    }
    set site_state[i][x].req_type = M; /* req_type initially set to migratory */
  }
}
```

Execution Rules

Each site i in the network is driven by the following events: i generates a Req_msg, i receives a Req_msg, i receives a DR_mig_msg and i receives an Ack_msg. It should be noted that each of these actions is processed without interruption. It is also assumed that all operations on a DR are atomic in nature.

Site i generates a Req_msg for DR x

Whenever site i intends to access DR x, it first checks to see if it has DR x with it; if so, it accesses DR x and performs operations on it (lines 3 and 4). If it does not hold DR x, it generates a request for DR x, adds its request to its DR_queue for DR x and forwards the request to site pointed by loc_ptr$[i][x]$ (lines 5 to 9). Site i then enters into a requesting state for DR x (line 10). A detailed procedure for generating a request follows.

```
/* This procedure is executed by site i when it generates a Req_msg for DR x */
if (migration)
  set site_state[i][x].req_type = M; /* migratory request*/                    (1)
else
  set site_state[i][x].req_type = A; /* access request */                      (2)
if (site_state[i][x].state==H)                                                 (3)
  DR x access();                                                               (4)
else /* site i does not have DR x, therefore it generates a Req_msg
and adds it to DR_queue[i][x] */                                               (5)
{
  create a Req_msg with site_id = i,
  DR_id = x, req_type = site_state[i][x].req_type;                             (6)
  set DR_queue[i][x].site_id = i;                                              (7)
  set DR_queue[i][x].req_type = site_state[i][x].req_type;                     (8)
  send Req_msg to loc_ptr[i][x]; /* site i forwards the request
  to the site given by its loc_ptr for DR x */                                 (9)
  set site_state[i][x].state = R; /* site i changes its state to requesting */ (10)
}
```

Site i receives a Req_msg of site j for DR x

Whenever site i receives a request message for DR x it enters into requesting state for DR x (line 11). If site i is holding DR x and the received request is migratory,

then DR x is migrated to site j and site i changes its state for DR x to not requesting (lines 12 to 15). The loc_ptr of site i for DR x is also changed to point to the requesting site (line 16). And in case site i is holding DR x and the received request is access, then site i sends an acknowledgement message to site j, thereby allowing site j to perform fixed processing on DR x, at site i (lines 18 and 19). Its state for DR x is changed to not requesting (line 20). On the other hand, if site i is not holding DR x, it puts the request of site j in its DR_queue for DR x (i.e. adds site_id of the requesting site (j) and req_type (migratory or access) in the DR_queue for DR x) (lines 21 to 23). This is because site i will receive DR x in a finite amount of time and would thus be able to fulfill the request of site j.

```
/* This procedure is executed by site i when it receives
a Req_msg of site j (Req_msg.site_id) for DR x */
set site_state[i][x].state = R;                                    (11)
if (site_state[i][x].state==H)                                     (12)
{
  if (Req_msg.req_type==M)                                         (13)
  {
    send DR_mig_msg to Req_msg.site_id; /* migrates DR x */        (14)
    set site_state[i][x].state = N;                                (15)
    set loc_ptr[i][x] = Req_msg.site_id; /* loc_ptr is set to the site
    where DR is migrated */                                        (16)
  }
  else /* received request is access */                            (17)
  {
    create an Ack_msg with site_id = i, DR_id = x;                 (18)
    send an Ack_msg to Req_msg.site_id;                            (19)
    set site_state[i][x].state = N;                                (20)
  }
}
else /* Add Req_msg to DR_queue[i][x] */                          (21)
{
  set DR_queue[i][x].site_id = j;                                  (22)
  set DR_queue[i][x].req_type = Req_msg.req_type;                  (23)
}
```

Site i receives a DR_mig_msg for DR x

This message is received by site i, in the following two cases: i) as a result of a previous migratory request generated by it; or ii) when site i had migrated DR x to some other site and now it receives the DR back. The following cases arise:

Case A: Site i is not the default site for DR x

This implies that DR x is received in response to the migratory request of site i. Site i enters into holding state for DR x and sets its loc_ptr[i][x] to point to itself (lines 24 and 25). After completing its operations on DR x, it returns DR x to

the default site of DR x and changes its state to not requesting. The loc_ptr of site i for DR x is also changed to default site (lines 26 to 33).

Case B: Site i is the default site for DR x and its DR_queue for DR x is not empty

Whenever site i receives this message, it enters into holding state for DR x and sets its loc_ptr$[i][x]$ to i (lines 24 and 25). Site i then checks its DR_queue for DR x, in case it is not empty, it indicates some requests are waiting to be satisfied. The entries in the DR_queue for DR x, at site i, are then read in FIFO manner. Depending upon the entries in DR_queue for DR x, the following cases need to be considered.

B.1: DR_queue$[i][x]$.site_id $= i$

This implies that it is site i' own request to access DR x. As site i now holds DR x, it can access DR x and perform operations on it (lines 36 to 38).

B.2: DR_queue$[i][x]$.site_id $= j$, $j \neq i$ and the req_type of site j is access

This implies that an access request of site j for DR x is waiting at site i. As site i now holds DR x, it sends an Ack_msg to site j. Site j can then perform operations on DR x at site i (lines 39 to 41).

B.3: DR_queue$[i][x]$.site_id $= j$, $j \neq i$ and the req_type of site j is migratory

DR x is migrated to site j. loc_ptr$[i][x]$ is updated to the requesting site where DR x migrates and site i enters into requesting state for DR x. The request type of site i for DR x is also set to migratory so that site i receives DR x back and the process further requests for DR x (lines 42 to 46).

Case C: Site i is in a holding state for DR x and its DR_queue for DR x is empty

This indicates that no requests are waiting at site i for DR x. Therefore, site i remains in holding state for DR x (lines 47 to 49).

```
/* This procedure is executed by site i when it receives a DR_mig_msg
for DR x */
set loc_ptr[i][x] = i;                                          (24)
set site_state[i][x].state = H;                                 (25)
if (site i is not the default site for DR x)                    (26)
{
  remove the entry from DR_queue[i][x];                         (27)
  if (DR_queue[i][x].site_id==i)                                (28)
    DR x access();                                              (29)
  send DR_mig_msg to default site for DR x;                     (30)
  set loc_ptr[i][x] = default site for DR x;                    (31)
  /* DR x migrates to the default site for DR x */              (32)
```

```
set site_state[i][x].state = N;                                          (33)
}
else /* if site i is the default site for DR x */                        (34)
while ((site_state[i][x].state==H) AND (DR_queue[i][x] not empty)) do     (35)
{
  remove the entry from DR_queue[i][x];                                  (36)
  if (DR_queue[i][x].site_id==i)                                         (37)
  {
    DR x access();                                                       (38)
  }
  else if (DR_queue[i][x].req_type==A)                                   (39)
    {
      create an Ack_msg with site_id = i, DR_id = x;                     (40)
      send an Ack_msg to DR_queue[i][x].site_id;                         (41)
    }
    else /* Migratory */                                                 (42)
    {
      set site_state[i][x].state = R;                                    (43)
      set site_state[i][x].req_type = M;                                 (44)
      send DR_mig_msg to DR_queue[i][x].site_id;                         (45)
      set loc_ptr[i][x] = DR_queue[i][x].site_id; /* loc_ptr is set
      to the site where DR is migrated */                                (46)
    }
}
if ((site_state[i][x].state==H) AND (DR_queue[i][x] empty))               (47)
{
  set site_state[i][x].state = H;                                        (48)
  set loc_ptr[i][x] = i;                                                 (49)
}
```

Site i receives an Ack_msg forDR x

Site i checks its DR_queue for DR x to see if this message is received in response to its access request for DR x; if so, it can now perform operations on DR x (lines 50 to 53). Site i then changes its state to not requesting as its request has been serviced (line 54).

```
/* This procedure is executed by site i when it receives an Ack_msg */
if ((DR_queue[i][x].site_id==i) AND (DR_queue[i][x].req_type==A))         (50)
{                                                                        (51)
  remove the entry from DR_queue[i][x];                                  (52)
  DR x access();                                                         (53)
  set site_state[i][x].state = N;                                        (54)
}
```

6 SIMULATION ENVIRONMENT AND RESULTS

In this section, the simulation environment used in this paper is first briefly given and then the results obtained from the simulations are discussed.

6.1 Simulation Environment

The simulation environment considered in this paper is similar to that used by Hara et. al. in [14]. In the system, 32 sites and 20 DRs have been assumed, unless otherwise stated. The simulations have been performed on a binary tree network topology. A binary tree with network depth k (number of sites = 2^k) has sites as leaves and ATM switches (Figure 3 a) at every other level. The simulations for the RMM component of the proposed architecture have been performed. As already explained, before sending a message to a remote site, the RMM at the current site must form a SVC connection with the remote default site if such a connection does not already exist. The SVC connection would be automatically broken if idle for 20 minutes. The transmission delay is assumed to be negligible for every link in the network. Formulae 3 and 4 given below are used to calculate the total time required for transmitting one message between two sites, which consists of two parameters: i) the time required for setting up the SVC connection called Channel Establishment (CE) delay (Figure 3 b) and ii) the time needed for the transmission of the message once the connection is made called Message Transmission (MT) delay (Figure 3 c).

Thus, if h is the hop count (the number of ATM switches a message passes) between the two communicating sites, p' the total processing delay at both sites when SVC does not already exist, d is the constant propagation delay between any two ATM switches or any site and ATM switch, R and r the constant route configuration time and the constant routing time respectively at an arbitrary ATM switch, then

$$CE(h) = p' + 2(h+1)d + h(r+R). \tag{3}$$

If during transmission p is the total processing delay at both sites when SVC already exists, then

$$MT(h) = p + (h+1)d + hr. \tag{4}$$

Hence, the Total Time (TT) required for making a SVC connection between two sites and transmitting one message between them via h hops is determined by

$$TT(h) = xCE(h) + MT(h) \tag{5}$$

$$\text{where} \begin{cases} x = 0; & \text{when SVC connection already exists between two sites} \\ x = 1; & \text{when SVC connection does not exist between two sites} \end{cases}$$

The network parameters considered in this paper are consistent with those given in [14] and are tabulated in Table 3.

The simulations are carried out in PARSEC (PARallel Simulation Environment for Complex systems), which is a C-based discrete event simulation language [20].

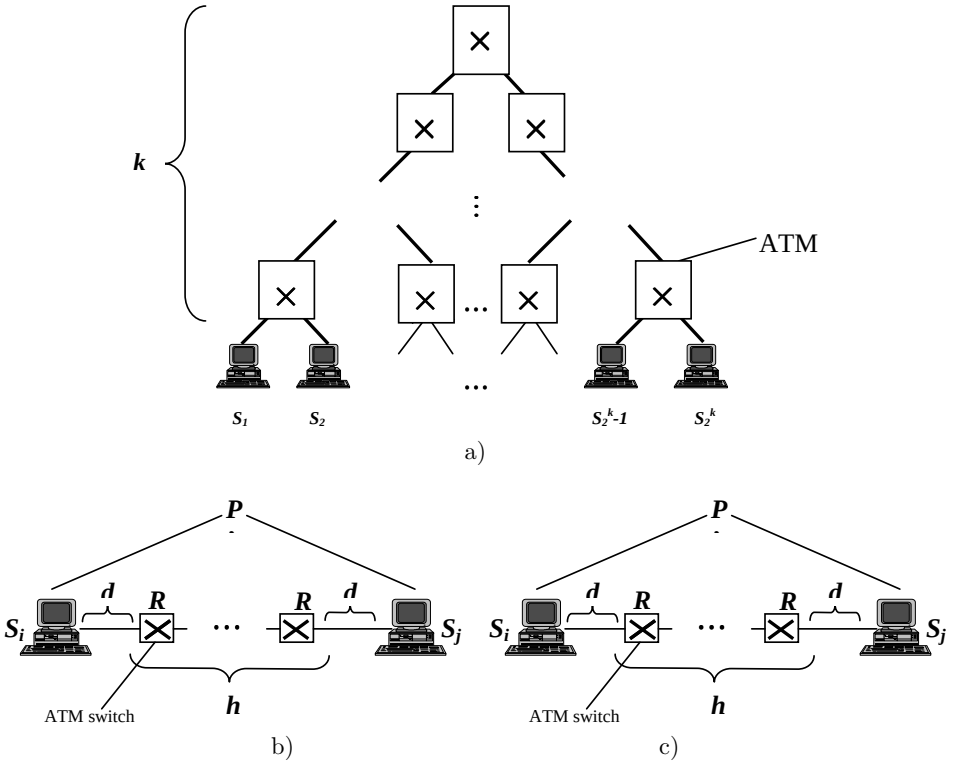


Fig. 3. a) Binary tree topology; b) Connection establishment; c) Message transmission

Parameters	values (seconds)
p	0.01
p'	0.03
r	0.002
R	0.01
d	0.005

Table 3. Network parameters

It adopts the process interaction approach to discrete event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes (events) are modelled by time-stamped message exchanges among the corresponding logical processes. All sites of the distributed system are simulated as entities, which have their local memory and communicate with each other via buffered message passing. Every entity is associated with a unique message buffer. Asynchronous send and receive primitives are provided by PARSEC to respectively deposit and remove messages from the message buffer of an entity.

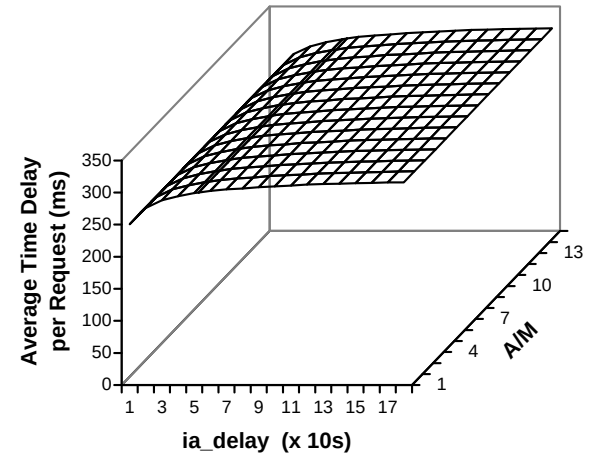
Time delay and message traffic are taken as the performance parameters for the evaluation of both the schemes (proposed and ECF). The time delay is the time required to locate and access the DR, which is the time spent between generation of the request and receipt of corresponding acknowledgement message or the requested DR. Message Traffic is the total number of messages (request, acknowledgement, and requested DR) generated by the system while processing access or migratory requests. These performance measures are probabilistic in nature, so therefore the values of these variables are collected for 5000 requests. The intervals between requests follow exponential distributions that are based on the mean access interval called inter access delay (ia_delay). The value of ia_delay is changed from 10 seconds to 3 minutes in the steps of 10 seconds. Both access and migratory requests are generated during simulation and the relative probability of generation of an access or a migratory request is proportional to Access to Migration (A/M) ratio, which is varied from 1 to 15.

Initially, each DR is assigned a default site such that no two DRs reside on the same site. The DR resides at its default site, when no migratory request is waiting for it. The loc_ptr table at every site is initialized to point to the default sites of each DR.

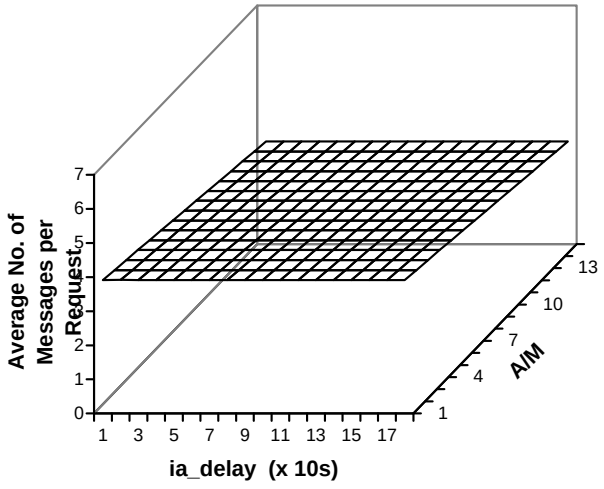
6.2 Simulation Results

In this section, the results obtained from various experiments are discussed. Figure 4 shows the results of the first experiment. In this experiment, ia_delay and A/M ratio are varied and the time delay is obtained using the proposed scheme for each pair of the values. This is shown in Figure 4 a where the x-axis indicates ia_delay, the y-axis indicates A/M ratio and z-axis gives the time delay. The values of time delay obtained varied from 250 ms to 315 ms. The message traffic is also measured through simulation experiments. As observed from Figure 4 b, message traffic declines from 3.91 to 3.01 with the increase in A/M ratio. It is concluded that the increase in A/M ratio and consequently the decrease in the number of migratory requests and hence in frequency of DR migration results in overall less number of messages.

In the second experiment, ECF scheme is simulated. Figure 5 a shows the value of time delay obtained by varying ia_delay and A/M ratio in the same manner as in the previous experiment. The time delay varies from 265 ms to 543 ms. Clearly the proposed scheme comes out to be far better than ECF in terms of time delay. It is observed that the proposed scheme gives lower average time delay values than ECF in every access interval and for all values of A/M ratio. This is because the DR does not return back to the site, from which it migrates and hence the request message has to travel a large number of sites in order to locate and access the DR. Further, it may be noted from Figure 5 b that the number of messages generated by ECF (5.3 to 4.2) is much higher as compared to the proposed scheme. This is because the requests are directly sent to the default sites in the proposed scheme, whereas, in ECF, more number of messages are required to find



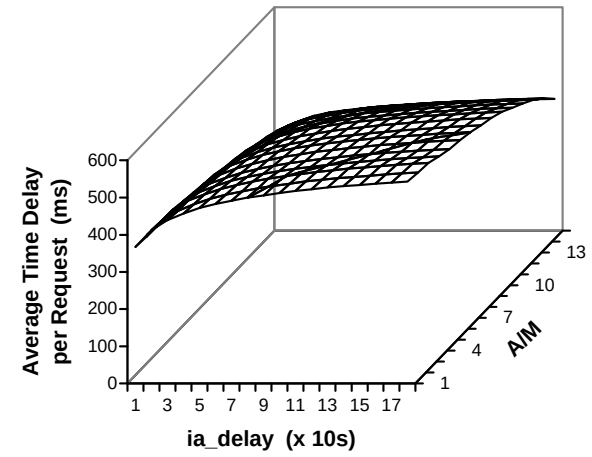
a)



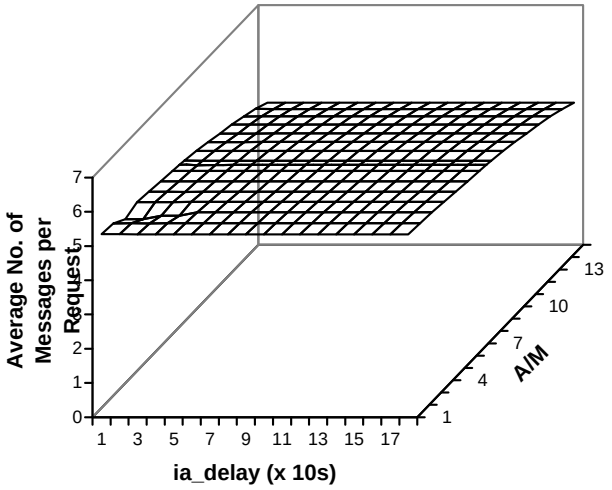
b)

Fig. 4. a) Time delay using the proposed scheme for binary tree topology; b) Message traffic using the proposed scheme for binary tree topology

the site holding the DR. The performance of ECF in terms of message traffic is inferior to the proposed scheme in every access interval and A/M ratio. It is also seen that the performance of ECF greatly deteriorates at lower A/M ratios. This is because at lower A/M ratios DRs migrate frequently which quickly renders their location information obsolete. Also many more messages are needed to update the information at sites about the location of DR on sites traversed by a request.



a)



b)

Fig. 5. a) Time delay using ECF scheme for binary tree topology; b) Message traffic using ECF scheme for binary tree topology

In the third experiment, the effect of varying network depth on time delay and message traffic using both the schemes is studied. For this, simulations for network depth equal to 6 and 7 for binary tree topology are performed. The number of DRs is kept fixed at 20. Figures 6 a and 6 b show the comparison of time delay and message traffic values using both the schemes for A/M ratio = 8. The time delay and message traffic values here are the average over the range of values of ia_delay considered previously. Figure 6 a shows that the performance of the proposed scheme in terms

of time delay is much better than ECF as the scale of network is increased. The graph shows that the time delay using ECF rapidly rises with the number of sites in the system. On the other hand the increase obtained by the proposed scheme is not as sharp. Network enlargement causes performance deterioration in ECF, possibly because, in case of ECF, the number of sites to which a DR can migrate increases; whereas in the proposed scheme, DR returns to its default site after satisfying the request. From Figure 6 b it is observed that in ECF message traffic increases nearly in proportion to the depth of the binary tree network; whereas the message traffic using the proposed scheme remains nearly unaffected with the depth of the binary tree network. The message traffic using ECF increases with network enlargement because, in ECF, the DR does not return to the site, from which it receives the DR and, hence, the request message has to travel a large number of sites in order to locate and access the DR.

In the fourth experiment, series of simulations are performed to study the effect of varying the number of DRs on time delay and message traffic in a binary tree network. Figures 7 a and 7 b give the time delay and message traffic using both the schemes for the DRs varying from 5 to 30. The number of sites in the network is kept fixed at 32. It is observed that the time delay given by the proposed scheme is always lower than that obtained by ECF scheme for all numbers of DRs. It is also noted that the number of messages using the proposed scheme remain constant as the DRs in the system increase, but the number of messages obtained by ECF increases with the increase in the number of DRs. Further, the number of messages obtained by the proposed scheme is always less than that obtained by using ECF.

Finally, it can be concluded from the above simulation results that the proposed scheme gives lesser time delay and message traffic as compared to ECF for all values of A/M ratios and ia_delay. This fact becomes more prominent as the scale of network and the number of DRs in the system increases.

7 CONCLUSION

In this paper, considering advanced features of ATM networks a new distributed system for transaction processing, DSTP-AN that is based on DR migration has been proposed. It attempts to achieve lower transaction processing times by providing a heuristic mechanism for the selection of the transaction processing method and by reducing the time delay in processing the requests to locate and access DRs. In DSTP-AN, the requests are not broadcasted to all the sites; instead they are propagated to the default sites. The requests therefore remain confined to a portion of the distributed system only. The performance of DSTP-AN, in terms of time delay and message traffic to locate and access the DRs is evaluated by a simulation study and is also compared with ECF. The simulation results show that in the proposed scheme the time delay and message traffic are reduced considerably in each access interval and A/M ratio when compared to ECF and this becomes more

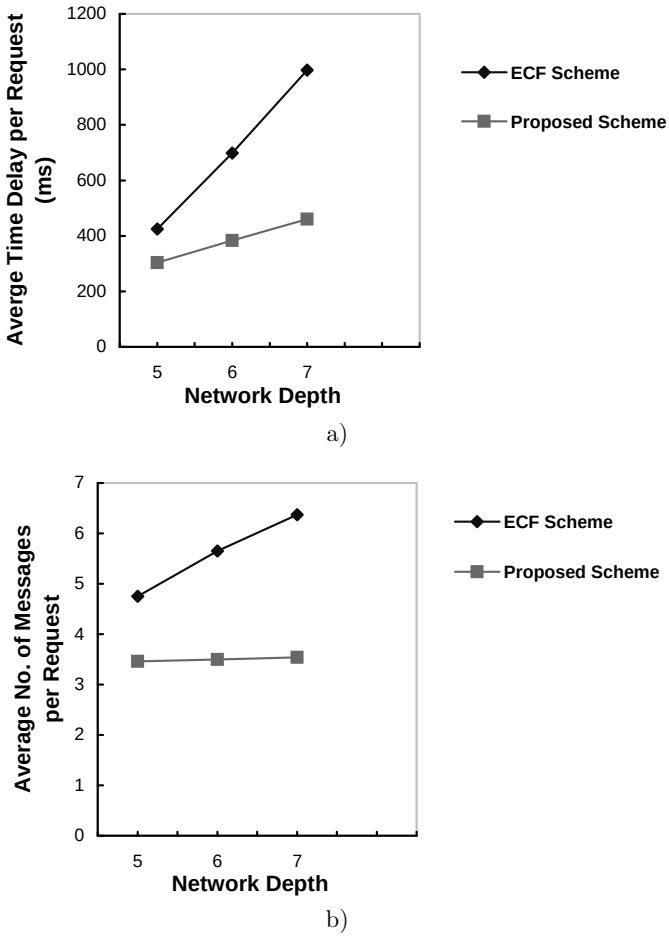
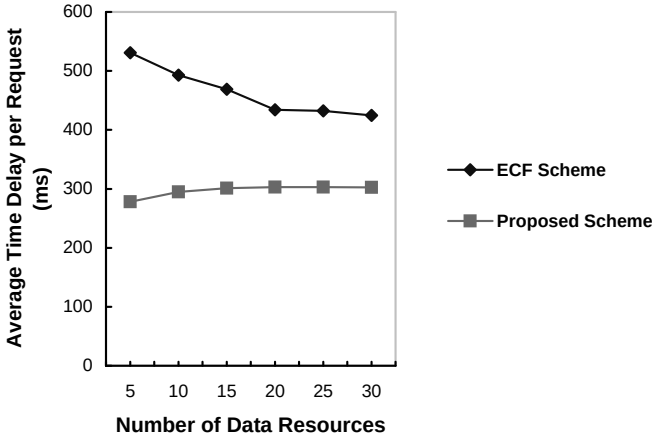
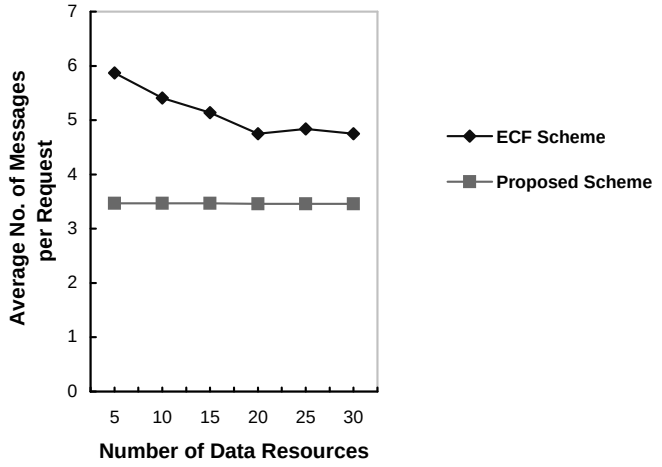


Fig. 6. a) Variation in time delay with respect to network depth for A/M = 8 and $m = 20$; b) Variation in message traffic with network depth for A/M = 8 and $m = 20$

obvious with the increase in network population and DRs. Further, the proposed scheme unlike ECF ensures fairness in the system; that is, the requests generated earlier are catered to first. This is because the request is stopped at the default site (by entering it in the DR queue) instead of being forwarded. Also, the concurrency control mechanism, employed by DSTP-AN, is completely distributed that involves no broadcasts.



a)



b)

Fig. 7. a) Variation in time delay with the number of DRs for A/M = 8 and $n = 32$; b) Variation in message traffic with number of DRs for A/M = 8

Acknowledgements

The authors are thankful to the reviewers for their valuable comments. It would not have been possible for us to put the paper in its present form without their suggestions.

REFERENCES

[1] Armbruster, H.: The Flexibility of ATM: Supporting Future Multimedia and Mobile Communications. IEEE Communications Magazine, Vol. 33, 1995, pp. 76–84.

[2] Apers, P.—Hevner, A.—Yao, S. B.: Optimization Algorithm for Distributed Queries. IEEE Transactions on Software Engineering, 1983.

[3] Bernstein, P. A.—Chiu, D-M. W.: Using Semi-Joins to Solve Relational Queries. Journal for the Association for Computing Machinery, pp. 25–40, 1981.

[4] Banerjee, S.—Li, V. O. K.—Wang, C.: Distributed Database Systems in High-Speed Wide-Area Networks. IEEE Journal on Selected Areas in Communications, Vol. 11, 1993, No. 4, pp. 617–630.

[5] Chia, S.: The Universal Mobile Telecommunication System. IEEE Personal Communication, Vol. 30, 1992, No. 12, pp. 54–62.

[6] Callender, M. H.: Future Public Land Mobile Telecommunication Systems. IEEE Personal Communication, Vol. 11, 1994, No. 4, pp. 18–22.

[7] Chiu, D-M. W.—Ho, Y. C.: A Method for Interpreting Tree Queries into Optimal Semi-Join Expressions. ACM SIGMOD, pp. 169–178, 1980.

[8] Enslow, P. H.: What is 'Distributed' Data Processing System. IEEE Computer, pp. 13–21, 1978.

[9] Guha, A.—Pavan, A.—Liu, J.—Ajay, R.—Steeves, T.: Supporting Real-Time and Multimedia Applications on the Mercuri Testbed. IEEE Journal on Selected Areas in Communications, Vol. 13, 1995, No. 4, pp. 749–763.

[10] Gavish, B.—Sheng, O. R. L.: Dynamic File Migration in Distributed Computer Systems. Communications of the ACM, Vol. 33, 1990, No. 2, pp. 177–189.

[11] Hac, A.: A Distributed Algorithm for Performance Improvement Through File Replication, File Migration, and Process Migration. IEEE Transactions on Software Engineering, Vol. 15, 1989, No. 11, pp. 1459–1470.

[12] Herman, G.—Gopal, G.—Lee, K.—Weinrib, A.: The Datacycle Architecture for Very High Throughput Database Systems. Proceedings ACM SIGMOD, pp. 97–103, 1987.

[13] Handel, R.—Huber, M. N.—Schroder, S.: ATM Networks, Concepts, Protocols, Applications. Third Edition, Addison Wesley, 1999.

[14] Hara, T.—Harumoto, K.—Tsukamoto, M.—Nishio, S.: Location Management Methods of Migratory Data Resources in ATM Networks. Proceedings of the ACM Symposium on Applied Computing (ACM SAC '97), pp. 123–130, 1997.

[15] Hara, T.—Harumoto, K.—Tsukamoto, M.—Nishio, S.: 'DB-MAN': A Distributed Database System based on Database Migration in ATM Networks. Proceedings of the 14th International Conference on Data Engineering (ICDE'98), Florida, pp. 522–531, 1998.

[16] Hara, T.—Harumoto, K.—Tsukamoto, M.—Nishio, S.: Database Migration: A New Architecture for Transaction Processing in Broadband Networks. IEEE Transactions on Knowledge and Data Engineering, Vol. 10, 1998, No. 5, pp. 839–854.

[17] Hurley, R. T.—Yeap, S. A.: File Migration and File Replication: A Symbiotic Relationship. IEEE Transactions on Parallel and Distributed Systems, Vol. 7, 1996, No. 6, pp. 578–586.

[18] Johnson, T.—Krishna, P.: Lazy Updates for Distributed Search Structure. Proceedings of ACM SIGMOD '93, pp. 337–346, 1993.

[19] Litwin, W.—Neimat, M. A.—Schneider, D. A.: LH* – Linear Hashing for Distributed Files. Proceedings of ACM SIGMOD '93, pp. 327–336, 1993.

[20] Meyer, R.: PARSEC User Manual. Release 1.1, Computer Science Department, UCLA, Los Angeles, CA 90024, 1998.

[21] Matsliach, G.—Shmueli, O.: An Efficient Method for Distributing Search Structures. Proceedings of the 1st International Conference on Parallel and Distributed Information Systems (PDIS), 1991.

[22] Nussbaumer, J. P.—Patel, B. V.—Scaffa, F.—Sterbenz, J. P. G.: Networking Requirements for Interactive Video on Demand. IEEE Journal on Selected Areas in Communications, Vol. 13, 1995, No. 5, pp. 779–787.

[23] Norp, T.—Roovers, J. M.: UMTS Integrated with B-ISDN. IEEE Communication Magazine, Vol. 32, 1994, No. 11, pp. 60–65.

[24] Nishio, S.—Tsukamoto, M.: Towards New Multimedia Information Base in Broadband Networks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. 79-D-II, 1996, No. 4, pp. 460–467.

[25] Reiner, D.: Special Issue on Query Processing. IEEE Database Engineering, Sept. 1982.

[26] Rooholamini, R.—Cherkassky, V.: ATM-Based Multimedia Servers. IEEE Multimedia, Vol. 2, 1995, No. 1, pp. 39–52.

[27] Saxena, P. C.—Choudhury, D. R.—Gabrani, G.—Gupta, S.: A Scheme for Managing Resource Locations in Broadband Networks. Computer Standards and Interfaces, Vol. 23, 2001, No. 5, pp. 399–427.

[28] Saxena, P. C.—Gupta, S.—Gabrani, G.: A Location Chasing Algorithm for Migratory Data Resources in ATM Networks. International Journal of Information & Computing Science, Vol. 3, 2000, No. 2, pp. 1–28.

[29] Saxena, P. C.—Gupta, S.—Gabrani, G.: A Security Model to Support Data Resource Migration. to CSI Journal of Informatics, Vol. 30, 2000, No. 4, pp. 21–29.

[30] Severance, C.—Pramanik, S.—Wolberg, P.: Distributed Linear Hashing and Parallel Projection in Main Memory Databases. Proceedings of the 16th International Conference on Very Large Databases, pp. 674–682, 1990.

[31] Vingralek, R.—Breitbart, Y.—Weikum, G.: Distributed File Organization With Scalable Cost/Performance. Proceedings of ACM SIGMOD '94, pp. 253–264, 1994.

[32] Wong, E.—Katz, R. H.: Distributing a Database for Parallelism. ACM SIGMOD, pp. 23–29, 1983.

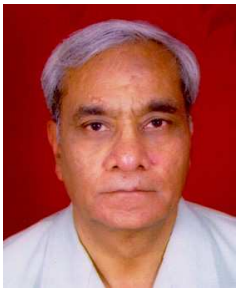
[33] Yu, C. T.—Guh, K. C.—Brill, D.—Chen, A. L. P.: Partitioning Relation for Parallel Processing Strategy in Fast Local Networks. Proceedings of the International Conference on Parallel Processing, 1986.

[34] Yu, C. T.—Guh, K. C.—Brill, D.—Chen, A. L. P.: Partition Strategy for Distributed Query Processing in Fast Local Networks. IEEE Transactions on Software Engineering, Vol. 15, 1989, No. 6, pp. 780–793.

[35] Coulouris, G.—Dollimore, J.—Kindberg, T.: Distributed Systems: Concepts and Design. Second Edition. Addison Wesley, 1994.

[36] Ramakrishnan, R.—Gehrke, J.: Database Management Systems. Second Edition. McGraw-Hill International Editions, Computer Science Series, 2000.

[37] Kadobayashi, R.—Tsukamoto, M.: Performance Comparison of Mobile Support Strategies. Proceedings of the 1st International Conference on Mobile Computing and Networking, Mobicom '95, pp. 218–225, 1995.



Prem Chandra Saxena is a Professor of computer science at School of Computer and System Sciences, Jawahar Lal Nehru University, New Delhi, India. He has supervised nine Ph.D. students in the area of Database Management, Networking and Multimedia. He has also guided 55 M.Tech. Dissertations. He has published several research papers in International and National Journals. His research interests include DBMS, Data Communication, Networking, Distributed Computing and Multimedia.

D. Roy Choudhury obtained M. Tech. degree from University of Calcutta, Calcutta 1966 and PhD. from the same university in 1971. From 1971 to 1973, he was associated with the Institute de Reglage Automatique EPFL, Switzerland. Since 1974, he has been in the Delhi College of Engineering, University of Delhi, New Delhi, India. At present, he is Professor in Computer Engineering Department at Delhi College of Engineering. His field of interest includes distributed systems, neural networks and petri nets.

Goldie Gabrani is an Assistant Professor in the Department of Computer Engineering at Delhi College of Engineering, University of Delhi, New Delhi, India. She received Bachelor, Master and PhD. in Engineering in the years 1984, 1990 and 2003, respectively. She has guided several B.E projects and around 25 ME Dissertations. She has published several research papers in international and national journals. Her research interests include distributed systems, networks and digital systems design.

APPENDIX

In ECF, whenever the requesting site generates a request message to access a DR, the request message is forwarded successively along the migration track of the requested DR, i.e. the chronological sequence of the sites at which the requested DR has resided. ECF scheme has a local location table at each site and does not have any DR queues. In ECF, the request messages are directed to the DR holding site using the local location tables maintained at each site. The local location tables are maintained by making the requests, DRs and acknowledgements carry the contents of local location tables that update the DR location tables at the sites they visit. The local location table at each site stores the latest known location of all the DRs along with their migration counts. The migration count of DR is increased by one each time the DR migrates from one site to another. Hence, the value i of migration count gives the corresponding location information of the DR after it has migrated i number of times. Hence, the newer information about the location of a DR can be obtained by comparing the migration count of a DR. Whenever a requesting site intends to access a DR, it sends a request message to a site according to its own local location table. If the site receiving the request message does not hold the requested DR, the site forwards the request message to another site according to its own local location table. The request message also carries the contents of local location table of the requesting site. At each site visited by the request message, migration counts of each DR given by the local location table of both request and the site are compared and older values are modified to newer values. This process of successive message forwarding continues till the request message reaches the site holding the requested DR. The holding site then sends the DR to requesting site (in case of migratory request). The entry regarding the current location of the requested DR in local location tables at both the holding and the requesting sites is then updated to requesting site and the migration count is incremented to the value one greater than that of DR holding site. And in case of an access request the DR holding site sends an acknowledgement message to the requesting site, thereby allowing it o access DR at it. No updation in local location tables at the DR holding site and requesting site takes place in this case. Moreover, ECF also uses another message called update message that is generated by the DR holding site whenever it receives a request message. Each update message follows the route taken by the corresponding request in reverse direction to the request generation site where it ends and updates the local location tables at all the sites falling in its way. It may be pointed out that the updation is performed in a similar way as by request message.