# Adlets Migration and Handling in a Distributed Object Environment

Hossam I. Gharib, Ping-Wen Chen, Yasuro Kawata and Shi-Kuo Chang
*Department of Computer Science*
*University of Pittsburgh, Pittsburgh, PA 15260*
*{hgharib, pwchen, yasuro, chang}@cs.pitt.edu*

## Abstract

*Large distributed systems may contain millions of objects that have to be managed. This makes it impossible to specify management policies for individual objects. Instead it is necessary to specify policy for groups of objects. This paper describes a framework for the construction of the **AD**lets **D**istributed **M**igration **M**echanism (**ADDMM**), which is used by the Adlet System for the management of adlets. The Adlet System [1,2] is a new approach for multimedia information retrieval and fusion based on the concept of active documents advertising, whereby an adlet (the metadata of a document) travels in the network to seek out documents of interest to the parent document, and at the same time advertises its parent document to other interested adlets. ADDMM provides the Adlet System with the facility of managing migrating adlets. ADDMM supports location-independent transparency and minimal interface by utilizing the ADDNS naming system [3] that provides a mechanism for transparent naming and locating adlets, which in turn supports ADDMM transparency at the adlet access level. With the ADDMM migration mechanism, various adlet migration protocols can be constructed and deployed. A visual diagram is introduced to specify such protocols for adlets migration, which can also be employed to specify distributed object management policies.*

**Keywords:** Object migration, distributed objects, information fusion, migration mechanisms, information retrieval, naming system.

## 1. Introduction

The advent of the World-Wide Web as an automated distribution mechanism for information has emphasized the need for tools to mediate between browsers, servers, and databases. The actual means to disseminate information widely is now reasonably cheap and reliable; the challenge is to find and tap into the information sources, integrate them, and provide useful responses to queries from an ever-larger, more demanding, yet ever less-specialized audience [10,13].

This shift from platform-dependent database and client/server architectures to Web-based storage and interaction over more heterogeneous data has increased the need for maintaining comprehensive descriptions of data structure, content, and other properties (metadata). It also increases the need for dynamic manipulation of both data and metadata [10,14,17,20].

One of the main research interests in distributed systems is multimedia information retrieval and fusion on the Internet. Chang and co-workers [1,2,3,18,19] present the formal definition of adlets approach that is based on the concept of active documents advertising**,** whereby the metadata of a document travels in the network to seek out documents of interest to the parent document, and at the same time advertises its parent document to other interested documents. The abstraction of metadata is called an adlet. In other words, it is the adlet, which travels, and performs information fusion and reporting, rather than the document or the end user. Being lightweight and adaptive, the adlet serves as an intelligent surrogate of the document and the end user. To accomplish this, one of the requirements of the Adlet System is to be able to handle, migrate, and maintain portable information for the adlets so that they can be retrieved no matter where they are.

Adlet migration is the relocation of an adlet from its current location (the source node) to another node (the destination node). Adlet migration is necessary so that adlets can travel freely in a distributed system [11,16]. The flow of execution of a migrating adlet is illustrated in Figure 1. An adlet may be migrated either before it starts executing on its source node or during the

course of its execution [4,5,6,15]. Adlet migration involves the following three major steps:
1. Selection of an adlet that should be migrated.
2. Selection of the destination node to which the selected adlet should be migrated.
3. Actual transfer of the selected adlet to the destination node.

The first two steps are taken care of by the adlet migration policy, and the third step is taken care of by the adlet migration mechanism.

The objective of this paper is to introduce a migration mechanism (ADDMM) that supports the developing concepts of the adlet in a way that facilitates the systems activities and abstractions in the distributed environment. ADDMM plays a very important role in achieving the goal of transparency, minimal interface, minimal residue dependencies, efficiency, and robustness [6,7,8].

This paper is organized as follows. Section 2 introduces the related work. Section 3 describes the ADDMM system architecture and adlet environment. Section 4 presents the ADDMM migration mechanisms. Section 5 defines the formal visual notation used to describe a set of adlet adaptive moving protocols. Section 6 introduces an example to illustrate the implementation. Concluding remarks and future research are given in Section 7.

## 2. Related work

Adlet migration shares some similarity with object migration but there are also important differences: object migration, in particular process migration, is often determined by the system to balance workload and therefore transparent to the application, while adlet migration is determined by migration policy preset by the application (or at least influenced by the application). [4,5,6,7,8,9,12,17,20].

## 3. The System Architecture

The ADDMM attempts to balance between different issues of handling and migrating Adlets. The architecture of the Adlet System is shown in figure 2. The ADDMM interacts with the Adlet Distributed Manager (ADDM) [2] to service the system and clients requests to perform the migration process.

An adlet is the means by which an active document makes itself known to other active documents. The documents are active in the sense that each document can perform certain actions with autonomy. An Adlet is defined as *Adlet= (doc-id, type, profile, target, non-target, ad-strategy, prop)* with an identifier, a type, a profile, a target set of documents, a non-target set of documents, an advertising strategy and additional properties [1,2].

The local distributed ADDNS system is an extension to the Adlet Distributed Manager (ADDM). The Global ADDNS name sever is a remote naming server that combines the local databases of the local ADDNS naming system plus additional information on the state, new locations and others of the adlets. This concept reduces the system overhead and also the number of messages passing between different nodes [3].

The *adlet environment* is defined as Env (V, L, $A_T$, $F_{primitive,}$ $S_{ph}$), where
V = {n1, …., $n_{nv}$,} is the set of nodes in the system,
L = { ($n_1$, $n_2$) , … , ($n_i$, $n_j$) } is the set of the links between nodes,
F = { $F_{move}$ , $F_{delete}$, $F_{copy}$, …, $F_{suspend}$} is the set of primitive functions,
$A_T$ = { $at_1$, $at_2$, $at_3$, $at_4$, ..., $at_{na}$ } is the set of adlet types in the system,
$S_{ph}$ = { ($n_1$, $at_1$), ($n_1$, $at_2$), …, ($n_i$, $at_j$)} is the set of occurrence of an adlet type in different nodes of the system.

The *locating adlet* Loc($n_i$, $ad_i$) = $S_{ph}$ = {($n_1$, $at_1$), ($n_1$, $at_1$), …, ($n_i$, $at_j$)} where $ad_i$ is the Adlet name. The set of $S_{ph}$ defines the occurrence of the adlet $ad_i$ in altered nodes with different types. The Adlet System should apply selection criteria to select the migrated adlet.

The *adlet migration process* mig($ad_i$ , $n_{current}$, $n_{new}$ ) = $S_{ph}$ - { ($ad_i$ , $n_{current}$ )} + {($ad_i$ , $n_{new}$ )} where M is the *move* function and D is the *delete* function.

## 4. ADDMM Migration mechanism

Migration of an adlet is a complex activity that involves proper handling of several sub-activities in order to meet the requirements of a good object migration mechanism. The three major subactivities involved in adlet migration are as follows.
1. Freezing the adlet on its source node and restarting it on its destination node.
2. Transferring the adlet's data space from its source node to its destination node.
3. Forwarding messages meant for the migrant adlet.

In the following, we will define mechanisms for handling each of these subactivitics.

### 4.1 Mechanism for Freezing and Restarting an Adlet

In adlet migration, the usual adlet is to take a "snapshot" of the adlet's state on its source node and reinstate the snapshot on the destination node. For this, at some point during migration, the adlet is frozen on its source node, its state information is transferred to its destination node, and the adlet is restarted on its destination node using this state information. By freezing the adlet, we mean that the execution and activities of the adlet is suspended and all external interactions with the adlet are deferred. The mechanism is defined as following

1. Check the status of the adlet
2. If the adlet is not performing any action or activity, it can be immediately freeze from further execution.
3. If the adlet is performing any action or activity, the freezing may have to be delayed until the adlet action or activity is complete.
4. If the adlet is performing any data retrieval, data storage, or search, the freezing may have to be delayed until the adlet action is complete.

### 4.2 Data Space Transfer Mechanism

The migration of an adlet involves the transfer of the following types of information from the source node to the destination node

- Adlet's state
- Adlet's data space

The mechanism is defined as following

1. Check the status of the adlet.
2. If freeze, create a copy of the adlet's state and data space.
3. Send the adlet copy to the destination node.
4. The newly created adlet should have the same adlet identifier as the migrating adlet.
5. Check the status of the adlet copy on the destination node.
6. If the adlet copy is Okay, then Update the new location information in the Global Database naming system.
7. Delete the source adlet information from the local Database naming system of the source node.
8. Set the source/copy flag to source of the adlet ID on the destination node.

### 4.3 Message-Forwarding Mechanism

In moving an adlet, it must be ensured that all pending, en-route, and future messages arrive at the adlet's new location. The messages to be forwarded to the migrant adlet's new location can be classified into the following

Type 1: Messages received at the source node after the adlet's execution has been stopped on its source node and the adlet's execution has not yet been started on its destination node

Type 2: Messages received at the source node after the adlet's execution has started on its destination node

Type 3: Messages that are to be sent to the migrant adlet from any other node after it has started executing on the destination node.

Our mechanism to handle these types of messages is very simple and straightforward. Because when the system directs these messages to the ADDNS naming system, the ADDNS system will check first the local Naming system in the source node. If not found, the ADDNS will automatically consult the Global naming system which in turn will direct the messages to the destination node [3].

## 5. Adlet Adaptive Moving Protocols

In defining and designing the adlet protocols, we employee a formal visual notation that is constituted with the *Itinerary Diagram (I-Diagram)*. The Itinerary Diagram is used to visually present an information activity protocol in the adlet System. An Itinerary Diagram can be formally denoted by a 7-tuple (V, T, N, O, R, I, s, f), where

• V is the set of virtual sites. In each virtual site, there exists adlets. In the diagram, an ellipse presents a virtual site.

• T is the set of adlet types. Each adlet is an instance of an adlet type. An adlet type is textually represented.

• N is the set of nodes. Each node denotes an adlet of a type from T. A node is represented by a circle inside a virtual site.

• O is the set of operations. Each operation belongs to one of the following types: SEND, SEND&SUICIDE, CREATE, CREATE&SUICIDE, CLONE, CLONE&SUICIDE, and BE (where BE is either "become" or "physical migration").

The visual presentation of each operation type is defined in Table 1. An operation is represented by an arc, the visual presentation of its operation type, associated with a textual label to describe this operation.

• R is the set of directed relationships between two adlets in N. R is a subset of V ✕ N ✕ 0 ✕ V ✕ N. (vl, nl, o, v2, n2) denote each element in R. It represents that adlets nl and n2 are located in virtual sites vl and v2, respectively, and there is a directed relationship from nl to n2 through operation o.

• I is the set of adlets which are loaded with collected information. $I \subset N$. Each Adlet in I is visually presented by a filled node.

• s is the initial adlet, $s \in N$. The initial adlet is the adlet submitted by the user to start the information retrieval; a node with an arrow visually presents s.

• f is the final adlet. $f \in N$. The final adlet is the adlet responsible of the termination of the information retrieval protocol. A double circle visually presents f.

The following set of adlet protocols are descried and illustrated using the formal visual notion defined in Table 1.

### 5.1 Parallel Protocol (PP)

In this type of protocol, the Adlet Distributed Manager (ADDM) [2] forwards a special search message from the client to a Master adlet (see figure 3). The Master adlet creates n number of Scout adlets to handle the mission. Each one of the Scout adlet will aim to search in different site. The adlet Master copies (Scouts) have the same type but with different ID. The ADDNS adlet naming system [3] keeps track of these copies and update their status instantly. In each site, the Scout Adlet will start a search operation and at the end of this operation will "Become" a loaded Scout adlet. The shaded circles denote that the Scout adlet is loaded with information as a result of the searching process.

When each Scout adlet finishes searching in its site, the loaded Scout Adlets will return back the collected information to the Master adlet and suicide. The ADDNS adlet naming system will remove the Scout adlets entries and their information from the naming system database. The Adlet Distributed Manager by turn will issue a merge operation. The merge operation merges the collected information from different sites. This merge operation is applied to all copies of the Master adlet as shown in Figure 3.

### 5.2 Reconnaissance Protocol (RP)

The Reconnaissance Protocol (RP) describes a different behavior of the adlet system. The Master adlet creates only one adlet copy (Scout) to handle the mission as shown in Figure 4. This Scout adlet firstly send a message to the next site to create a new scout of the same type but with different ID to handle the search operation on this site. Secondly, the Scout adlet issues a local search operation on his site and if found it will "Become" a loaded Scout adlet. When the Scout adlet accomplishes the local search operation, the information is sent back to the Master adlet and the Scout adlet suicides. The created Scout of the next site will continue the same behavior until the end of the RP protocol.

In the RP protocol as illustrated in figure 4, we notice that the Scout adlet never travels from one site to another loaded with information, also each Scout adlet is assigned to perform the search operation in his site only, and it instructs another Scout adlet to complete the assignment on the next site and so on.

### 5.3 Reconnaissance Migration Protocol (RMP)

The Reconnaissance Migration Protocol (RMP) is unlike the RP protocol, it is the same adlet that migrates from one site to another instead of creating a new adlet in each site as in the RP protocol. As shown in figure 5, the Adlet Distributed Manager (ADDM) [2] forwards a special search message from the client to a Master adlet. The Master adlet by turn dispatches an adlet to perform the operation. This adlet carries out the following operations

1- Clone another adlet with the same type but with different ID to perform the local search operation.

2- Migrate to the next site.

As shown in Figure 5, we use sequence numbers on the output arcs leaving the nodes to define the order of the performed operations in the Itinerary diagram.

The clone adlet will issue a local search operation on his site. After the retrieve operation is accomplished, the clone adlet sends the retrieval information to the Master adlet and then suicides. The original dispatched adlet will migrate to the next site to continue the global search operation. The ADDNS naming system [3] will update the location information to reflect the new status of the migrated adlet.

The migrated adlet will act the same way on the new site and clones again an adlet to perform the search operation on this site. The migrated adlet will continue to migrate to the next site till it reaches the final site. When the migrated adlet reaches the last site, it performs the search operation, send the collected information to the Master adlet, and suicides. The Master adlet merges all the collected information from different sites. The Adlet Distributed Manager (ADDM) sends back the collected information to the requesting client.

### 5.4 Free Trip Protocol (FTP)

The Free Trip Protocol (FTP) defines a searching operation where the adlet searches different sites without

any time limitation. In other words, the adlet is free to travel from one site to another hoping to gather more and more targets without the time or size restriction. This operation is flexible and only concerns about the target information. The Free Trip operation could be considered as an unlimited depth searching level. The time or size of collected information is not limited or determined.

As shown in figure 6, the Adlet Distributed Manager (ADDM) [2] send a search message, received from a client, to apply the FTP protocol to a Master adlet and also sets the time parameters $\phi$ to unlimited (see figure 6). The Master adlet creates a Scout adlet to handle the operation. The Scout adlet searches the local site for the desired information, send a clone request to the next site, and then destroys itself. The Clone adlets accumulate the collected information from one site to another. At the final site, the Clone adlet send all the collected information to the Master adlet. The ADDM by turn directed the collected information to the requesting client.

### 5.5 Free Trip Migration Protocol (FTMP)

The Free Trip Migration Protocol (FTMP) is the same as the FTP protocol except that the adlet itself migrates from one site to another loaded with the accumulated collected information (see figure 7). The ADDNS naming system [3] maintains the updated new location information of the migrated adlet.

### 5.6 Limited Trip Protocol (LTP)

The Limited Trip Protocol (LTP) is a limited searching operation, where the time factor is substantial to this operation. The client sends a limited search request of type LTP to the Adlet Distributed Manager (ADDM). The time parameter $\phi$ is also predetermined by the requesting client or by the adlet system defaults if the client omit it. As shown in Figure 8, The adlet searches and travels from one site to another. When the specified time $\phi$ is due, the adlet returns back to the owner site with the available collected information. The LTP protocol is useful when the client is concerned only with the response time of searching process.

### 5.7 Limited Trip Migration Protocol (LTMP)

As shown in figure 9, The Limited Trip Migration Protocol (LTMP) acts differently from the Limited Trip Protocol (LTP). The dispatch Clone adlet performs by itself a local search operation on the owner site, checks the time parameter $\phi$ due time, and if $\phi$ not due: migrates with the search result to the next site. The same set of actions and operations are performed in each site until the $\phi$ time parameter is true. When the $\phi$ is due, the migrated adlet stops the trip, send the collected information to the Master adlet, and suicides. The ADDNS naming system deletes the migrated adlets record from the Naming Database. The Adlets Distributed Manager (ADDM) forwards the collected information in the Master adlet to the requesting client.

## 6. Implementation

In our implementation of the experimental prototype of the new adlet protocols we employ the Adlets Distributed Migration Mechanism (ADDMM) to mange the moving actives of adlets in the distributed environment. Our approach is to illustrate the interaction of the Adlet Distributed Manager (ADDM), Adlet Distributed Naming Service (ADDNS), Adlets Distributed Migration Mechanism (ADDMM), and adlet Adaptive Moving Protocols through a given example of adlets in action. We select the Reconnaissance Migration Protocol (RMP) for this example (see figure 10 and the Appendix for detailed explanation).

## 7. Discussion

Adlet distributed migration mechanism (ADDMM) provides the following advantages for the system and the clients.

**Reducing average response time of adlets**. The average response time of the adlets of a node increases rapidly as the load on the node increases. Adlet migration facility may be used to reduce the average response time of the adlets of a heavily loaded node by migrating some of its adlets on a node that is either idle or whose adlet capacity is underutilized.

**Bettering system performance**. Adlet migration mechanism could be used for bettering system performance by forcing adlets to migrate to a node at which it has minimum turnaround time. The minimal turnaround time of nodes is determined by using the concept of *Test Adlet*. The Test Adlet is an adlet created by the system to measure and evaluate the turnaround time and performance of the nodes.

**Dynamic Load-balance of system resources**. The ADDNS naming system maintains information concerning the number of local and remote references to the adlets from each node. These information are employed to reveal to the rate of demanding and referring of each adlet by each ADDM in the system. The

ADDMM system utilizes these information to support decisions concerning load balance, improving performance, and forced migration of adlets.

**Utilizing resources effectively**. In the Adlets System, the capabilities of the various resources of different nodes are different. Therefore, depending upon the type of an adlet, it can be migrated to the most suitable node to utilize the system resources in the most efficient manner.

**Reducing network traffic**. Migration of an adlet closer to the resources it is using most heavily may reduce network traffic in the system if the decreased cost of accessing its favorite resources offsets the possible increased cost of accessing its less favored ones. In general, whenever an adlet performs data reduction (it analyzes and reduces the volume of data by generating some result) on some volume of data larger than the adlet's size, it may be advantageous to move the adlet to the location of the data. Another way to reduce network traffic by adlet migration is to migrate two or more adlets, which frequently communicate with each other, on the same node of the system.

**Improving system reliability**. Adlet migration facility may be used to improve system reliability in several ways. One method is to simply migrate a critical adlet to a node whose reliability is higher than other nodes in the system. Another method is to migrate a copy of a critical adlet to some other node and to execute both the original and copied adlets concurrently on different nodes. Finally, in failure modes such as manual shutdown, which manifest themselves as gradual degradation of a node, the adlets of the node, for their continued execution, may be migrated to another node before the dying node completely fails.

**Improving system security**. A sensitive adlet may be migrated and run on a secure node that is not directly accessible to general users, thus improving the security of the Adlets System environment.

We use the I-Diagram to specify the traveling protocols of the adlets system. In another research, we use the Distributed Rule-based Finite State Machine (DRBFSM) model to define the behavior of each type of adlets. However, it is tedious to check whether the specifications in both models are consistent or not, after the creation or modification of the specifications. Therefore, in our future work, we will investigate how to check the consistency of specifications between the I-Diagram and the DRBFSM model.

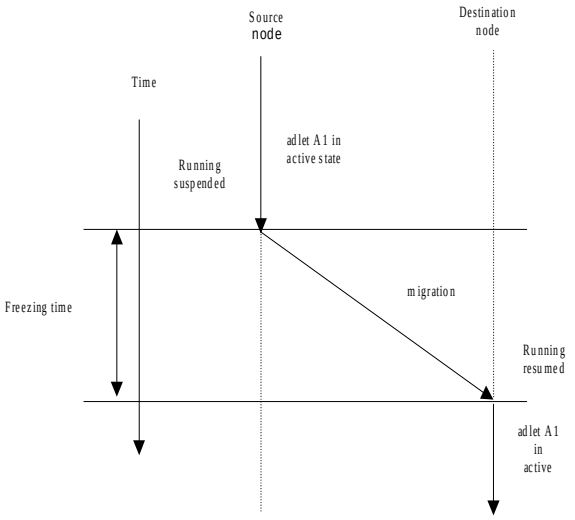
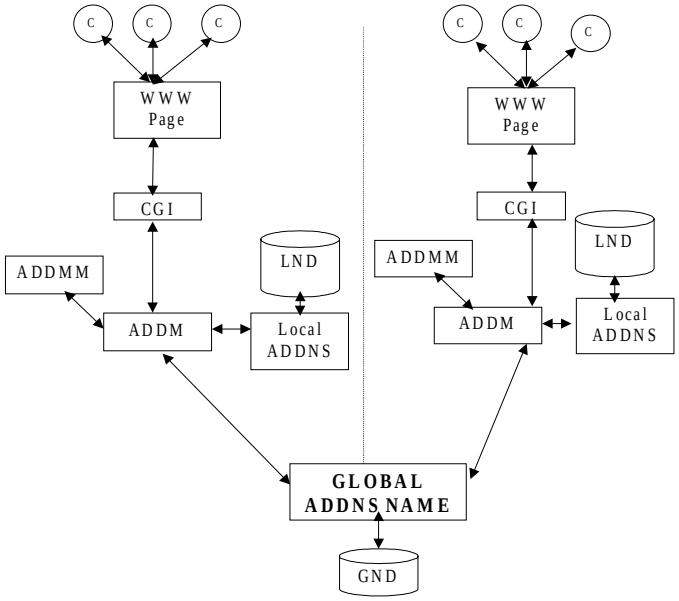



**Figure 1: Flow of execution of a migrating Adlet.**




ADDM: Adlet Distributed Manager.
GND: Portable Distributed Manager.
LND: Portable Local Naming Database. C: Client.

**Figure 2: The architecture of Adlet System.**

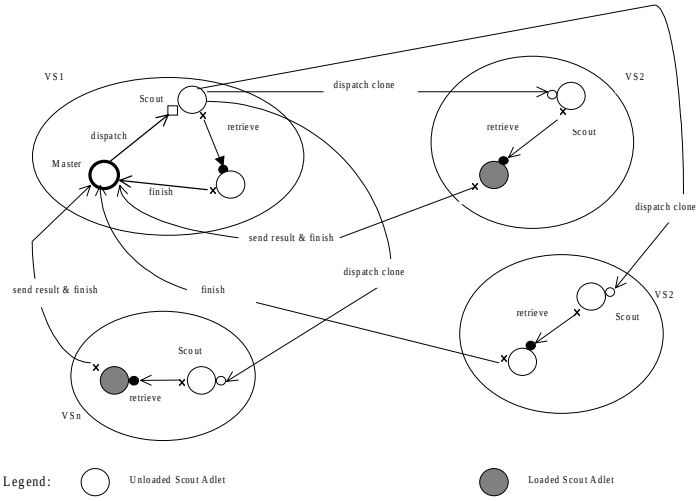



**Figure 3: Itinerary diagram for PP.**

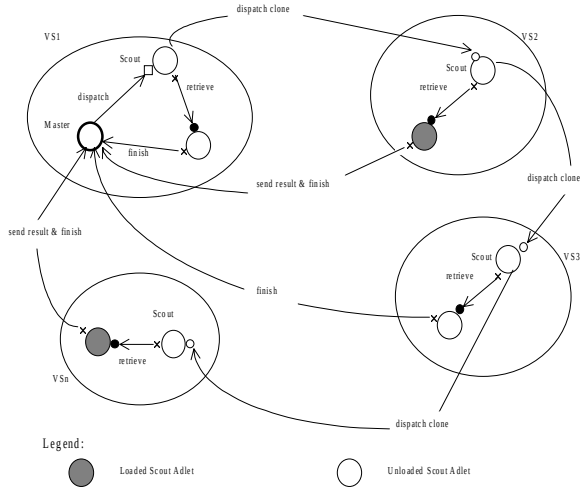



**Figure 4: Itinerary diagram for RP.**

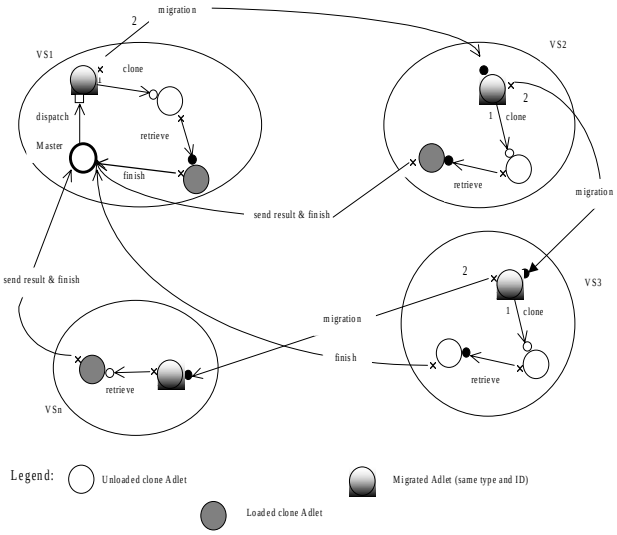



**Figure 5: Itinerary diagram for RMP.**

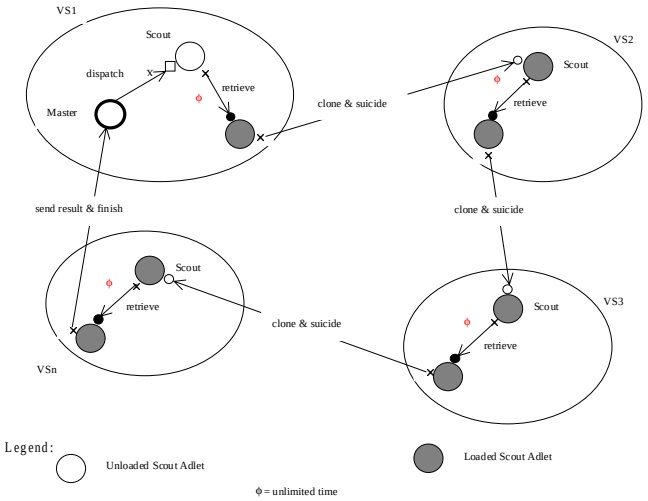



**Figure 6: Itinerary diagram for FTP.**

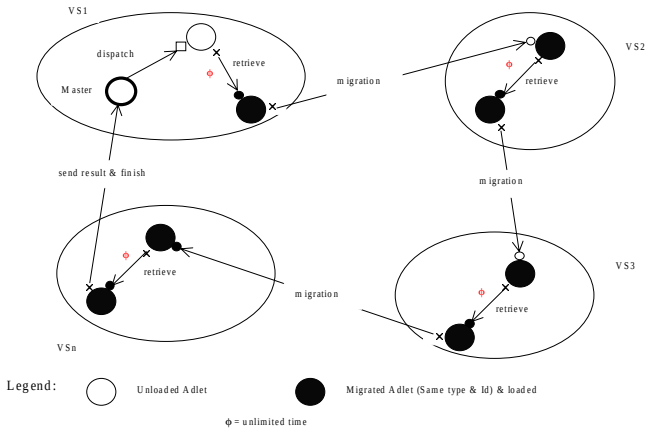



**Figure 7: Itinerary diagram for FTMP.**

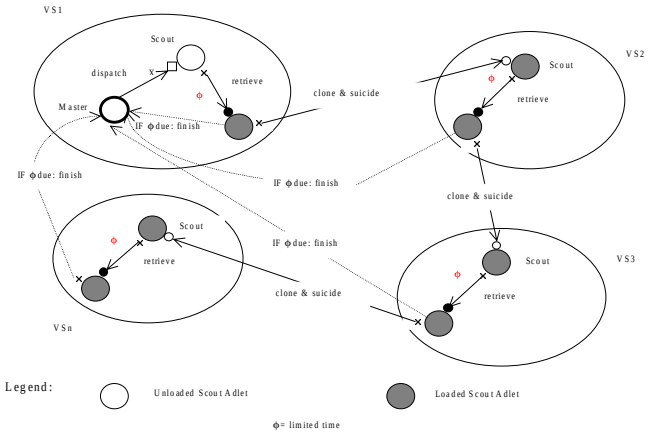



**Figure 8: Itinerary diagram for LTP.**

**Table 1: Types of arcs in I-Diagram.**

| Initiator \ Successor | | Creation: No | Creation: Yes — Different type ◇ | Creation: Yes — Same type, Different ID ○ | Creation: Yes — Same type, Same ID ● |
|---|---|---|---|---|---|
| Suicide | No | SEND → | CREATE →◇ | CLONE →○ | N/A |
| | Yes X | SEND&SUICIDE ✕→ | CREATE&SUICIDE ✕→◇ | CLONE&SUICIDE ✕→○ | BE ✕→● |

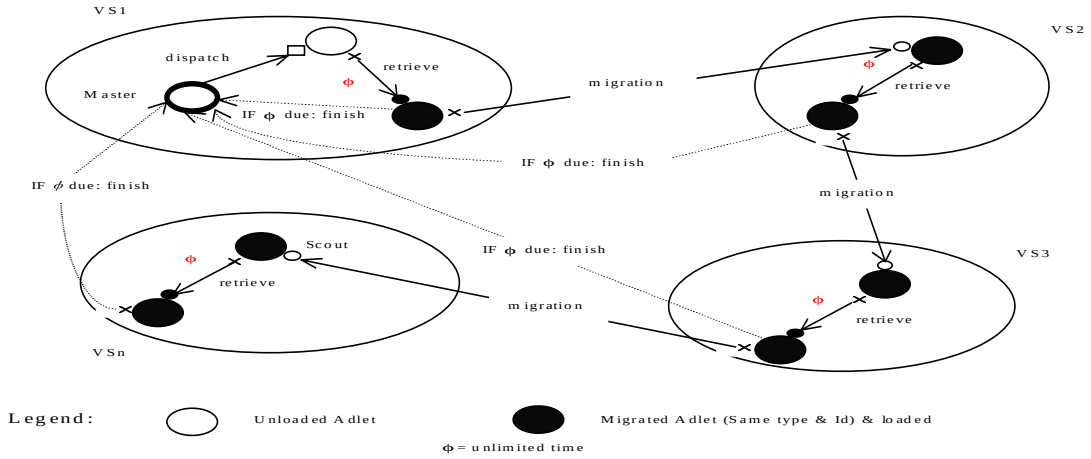



**Figure 9: Itinerary diagram for LTMP.**

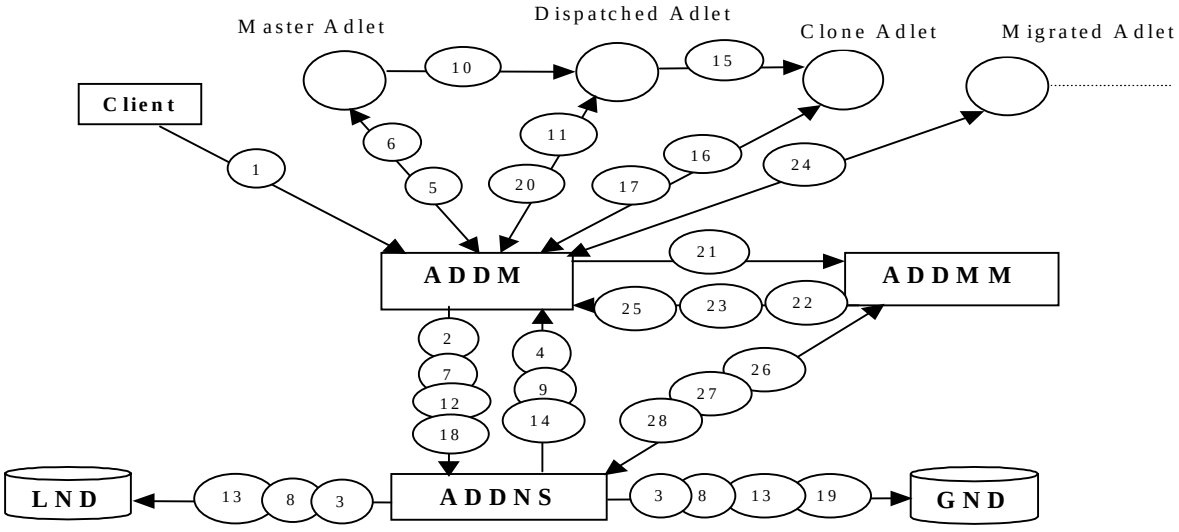



**Figure 10: Interaction of the ADDM, ADDNS, and ADDMM in the RPM protocol.**

**Appendix:** The following are the detailed steps of applying the ADDMM mechanism in RMP protocol (see Figure 10):

1. Request from the Client to search using RMP protocol.
2. Request from ADDM to ADDNS naming system to create and add Master adlet to the local naming database.
3. ADDNS maintains the Master adlet information in LND and GND.
4. ADDNS acknowledges to ADDM the Master adlet naming.
5. ADDM creates Master adlet.
6. Request from the Master adlet to ADDM to create a dispatched adlet with same type but different ID.
7. ADDM forwards the dispatched adlet creation request to ADDNS.
8. ADDNS adds dispatched adlet information in the LND and GND.
9. ADDNS acknowledges to ADDM the dispatched adlet naming.
10. ADDM creates dispatched adlet.
11.Request from the dispatched adlet to ADDM to create Clone adlet with same type but different ID.
12. ADDM forwards the dispatched adlet copying request to ADDNS.
13. ADDNS duplicates the dispatched adlet entries in LND and GND and set the Clone adlet source/copy flag to copy.
14. ADDNS acknowledges to ADDM the Clone adlet naming.
15. ADDM creates Clone adlet.
16. Request from the Clone adlet to ADDM to search the local site.
17. Request from the Clone adlet to ADDM to merge the collected information with the Master adlet and then suicides.
18. ADDM forwards the Clone adlet suicide request to ADDNS.
19. ADDNS deletes the Clone adlet record from LND and GND databases.
20. Request from the dispatched adlet to ADDM to migrate to the next site.
21. ADDM forwards the migration request to the ADDMM.
22. ADDMM consults ADDM for the status of the migrated adlet, IF not busy Then Freeze Else Wait.
23. Request from ADDMM to ADDM to create a copy of the migrated adlet's state and data space.
24. ADDM sends the copy with the same ID as of the migrated adlet to the next node.
25. Request from ADDMM to ADDM to check the status of the copy of the migrated adlet, IF OK Then continue Else repeat.
26. Request from ADDMM to ADDNS through ADDM to update the new location field of the migrated adlet in GND.
27. Request from ADDMM to ADDNS through ADDM to delete the migrated adlet information from LND of the local site.
28. Request from ADDMM to ADDNS through ADDM to set the Source/copy flag to source of the migrated adlet on the next site.
29. Repeat from 11.

**References:**


[1] S. K. Chang and T. Znati, "Fusion of Multimedia Information", Keynote Paper, Proceedings of the 4th International Workshop on Multimedia DataBase Management Systems (IW-MMDBMS'98), Aug 5-7, 1998, Dayton, Ohio, pp. 2-9.

[2] S. K. Chang, "Toward a theory of active index", Journal of Visual languages and Computing, 6(l): pp. 101-118,1995.

[3] H. Gharib, P. W. Chen, S. K. Chang, "Naming Service in the Distributed Object Environment for Adlets ( ADDNS )", IEEE Computer Society Press,1999.

[4] P. Sinha, Distributed Operating Systems: Concepts and Design, IEEE Computer Society Press, 1997.

[5] X. Jia et al., "Highly Concurrent Directory Management in the GALAXY Distributed System", Proc. l0th Int'l Conf. Distributed Computing Systems, IEEE Computer Soc. Press, Los Alamitos, Calif., 1990, pp. 416-423.

[6] M.L. Powell and B.P. Miller, "Process Migration in DEMOS/MP", Proc. Ninth Symp. Operating Systems Principles, ACM Press, New York, N.Y., 1983, pp. 110-119.

[7] K. Shimizu, M. Maekawa, and J. Hamano, "Hierarchical Object Groups in Distributed Operating Systems", Proc. Eighth Int'l Conf. Distributed Computing Systems, IEEE Computer Soc. Press, Los Alamitos, Calif., 1988, pp. 18-24,

[8] M.M. Theimer, K.A. Lantz, and D.R. Cheriton, "Preemptable Remote Execution Facilities for the V-System", Proc. 10th Symp. Operating Systems Principles, ACM Press, New York, N.Y., 1985, pp. 2-12.

[9] B. Walker, G. Popek, R. English, C. Kline, and G. Thiel, The LOCUS Distributed Operating System, In: Proceedings of the 9th ACM SIGOPS Symposium on Operating Systems Principles, Association for Computing Machinery, New York. NY. pp. 49-70, 1983.

[10] J. Crowcroft, Open Distributed Systems, Artech House Publishers, 1995.

[11] B. Verghese, S. Devine, A. Gupta, and M. Rosenblum, "Operating System Support for Improving Data Locality on CC-NUMA Compute Servers", ACM Press, MA, USA, 1996.

[12] P. K. Sinha, M. Maekawa, K. Shimizu, X. Jia, H. Ashihara, N. Utsunomiya, K. S. Park, and H. Nakano, The Architectural Overview of the Galaxy Distributed Operating System, IEEE Computer Society Press, 1994.

[13] S. Mullender, Distributed Systems, ACM Press, 1989.

[14] J. Wu, Distributed System Design, CRC Press, 1999.

[15] W. Stallings, Operating Systems, Prentice Hall, 1995.

[16] L. Wilson and W. Shen, "Experiments in Load Migration and Dynamic Load Balancing in SPEEDES", Proceeding of the 1998 Winter Simulation Conference, 1998.

[17] G. Weiss, Multiagent Systems, The MIT Press, Cambridge, Massachusetts, 1999.

[18] Ping-Wen Chen, Yasuro Kawata, Hossam I. Gharib, and Shi-Kuo Chang, "An Approach for the Design and Simulation of Information Retrieval Protocols", Proceedings of ICDCS International Workshop on Distributed System Validation and Verification (DSVV2K), IEEE Computer Society Press, Taipei, Taiwan, April 10-13, 2000.

[19] Yasuro Kawata, Ping-Wen Chen, Hossam I. Gharib, and Shi-Kuo Chang, "Managing Combination Products with Adlet in E-Commerce", Proceedings of IAT99 Workshop on Agents in Electronic Commerce, Hong Kong, China, December 14-17, 1999.

[20] J. Hu and M. P. Wellman, "Online Learning about Agents in a Dynamic Multiagent System", Proceeding of Autonomous Agents, Minneapolis, MN, USA, 1998, pp. 239-246.