An Immediate Messaging-based Multi-agent System using Linda Coordination Model

Sushil Kumar Gangwar
Student, Galgotias University
U.P India

Ritu Sindhu,Ph.D
AP, Galgotias University
U.P India

ABSTRACT
The quick development of mobile agents, the coordination of mobile agent becomes an up-and-coming technology in multi-agent structure for internet and distributed applications. This time an immediate messaging is getting popular and many people apply it to coordinate with each other in domestic as well as commercially. In this review paper I used a new approach for immediate messaging based Multi-agents using Linda like Coordination model. By this we have avoid the limitation of time and space. The Linda like coordination model having qualities of easy to implement and ubiquitous access. An ubiquitous meeting room and an immediate messaging mail service are provided in our structure for multi-agents to synchronize with each other in any place and at any moment. Some other qualities of Linda like coordination model i.e. it creates heterogeneous environment, platform independent and making useful application in object oriented programming because of all it, this creates well-designed interface for coordination of mobile agents.

General Terms
Trying to increase the efficiency of space and time, to be more specific, spatially uncoupled and temporally uncoupled conditions, for immediate messaging based on multi-agents using Linda like coordination model.

Keywords
Multi-agent Coordination, messaging, multi-agent structure, Linda Coordination model.

1. INTRODUCTION
The concepts of multi-agents [5] have been widely used in modern years. A mobile agent has the ability of organization, as well as mobility. The mobility and organization characteristics enable a mobile agent to cooperate with other multi-agents [3-11]. Thus, a Coordination model is needed in a multi-agent system for agents to coordinate with each other [8,6,1]. One of the most important things for mobile agent Coordination is to avoid the limitation of space and time, to be more specific, spatially uncoupled and temporally uncoupled conditions. The spatially uncoupled condition means the multi-agents do not need to coordinate in an appointed place. The temporally uncoupled condition means the multi-agents do not need to coordinate in an appointed time. A good Coordination model should achieve both spatially and temporally uncoupled conditions. Presently, based on the degree of these two conditions, there are four mobile agent Coordination models: direct, meeting-oriented, blackboard-based, and Linda-like Coordination models [2,7].Among these models, the Linda-like Coordination model can solve both the spatially coupled and temporally coupled problems. Furthermore, the approach can coordinate not only with another agent but also with real people. In these day, The business market of messaging is very large. There are many popular messaging systems, such as MSN messenger, Yahoo messenger and other messenger. In order to support our concept of using messaging to coordinate, we have conducted an experiment of the utilization of message tools for our students. With rapid development of the Internet and wideband services becoming popular, many students frequently stay on the Internet. That means most of the students can be found on the messaging server easily and multi-agents can finish the coordinating mission through our system.

2. RELATED WORK
There are many important research issues related to our work, such as mobile agent Coordination models, mobile agent systems, and messaging systems, etc. Mobile agent Coordination models are important components for mobile agent cooperation. Presently, mobile agent Coordination can be divided into four models [7]. The first model is the direct Coordination model. It is simple and some multi-agent systems use the direct Coordination model, such as Aglets etc. The second model is meeting-based, where multi-agents coordinate in a meeting room. The third model is blackboard-based. It is message board in which a coordinator writes a coordinating data in a blackboard, then the receiver goes to this blackboard to take the coordinating data. The last model is Linda-like. It uses a tuple space to build the Coordination share space. All tuple space will keep information identically. The logic name and some operations, such as Take, Write, Read, ReadAll, and TakeAll are used when agents access the tuple space. The Linda-like Coordination model is supported by Pagespace [2], Tucson [2], MARS [8, 10], etc. There are some other meeting-oriented mobile agent systems, such as Mole [14] and Ara (Agents for remote action [13]). The Mole mobile agent system has a build-in meeting room, The multi-agents can use Mole mobile agent system to coordinate with each other by sending an agent qualifier and a location name. The multi-agents do not have to care about the information of peer location. The Mole system helps the multi-agents finish coordinate operations. In addition, Ara mobile agent system also supports the meeting-based Coordination model. The system provides two communication methods for multi-agents. One is the service point and the other is the tuple space. The service point method transforms a mobile agent to a server to provide a meeting room which is a shared space for multi-agents to coordinate with each other. Mole and Ara provide meeting rooms, which are not omnipresent meeting rooms.

3. LINDA COORDINATION MODEL

Linda [12] is a communication model for an Immediate Messaging-based Multi-agent Coordination[4] System. It is suitable for a broad range of programming styles, e.g. the master-worker technique. It is always fixed inside a host language which provides the ability to do calculation. The Linda model provides communication between diverse processes involved in a computation. The communication is provided using a shared tuple space, which can be consideration of as a shared bag into which tuples are inserted and withdrawn by the user processes. The tuple space acts as a logical shared memory with an associative lookup method. To solve this we have to need some sort of kernel which acts as a controller of the tuple space. User processes then communicate with the kernel, sending it commands which represent the Linda primitives and tuples. In Linda a message between two tasks is never exchanged directly. A task that wants to output data, puts it into the tuple space, and a task that wants to read data, searches for it and reads it from the tuple space. Tasks communicating in this way need not be known to each other. Linda is a set of high-level operations that can be added to a base language to give way a parallel talk of that language.

Linda's unusual features make the language redolent and interesting in its own right. Where most distributed languages are partially distributed in space and non-distributed in time, Linda is fully distributed in space and distributed in time. Linda Multiple local tuple spaces are mainly used in distributed as well as parallel communication.

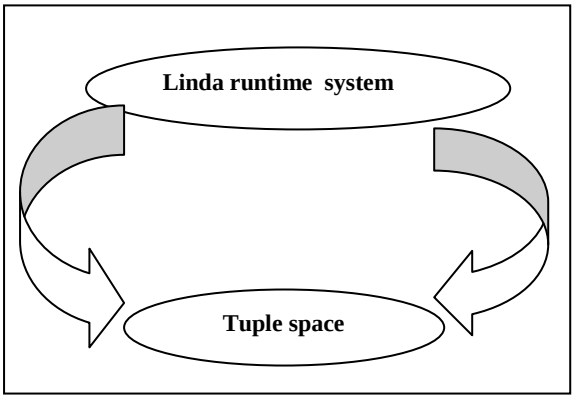

Figure 1: Single Local Tuple Space

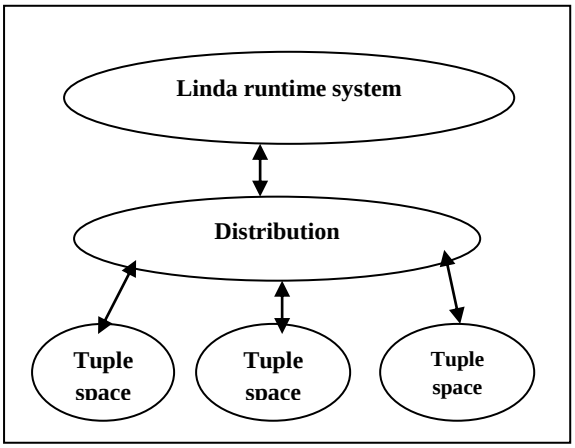

Figure 2: Mutiple Local Tuple Spaces

4. LINDA PRIMITIVES

The Linda model uses the exchange of tuples as the means of process Coordination. A tuple is an ordered sequence of heterogeneously typed objects. For example, the tuple [1, false,"Kavi"] contains an integer, a Boolean value and a string. When a tuple is to be retrieved from a tuple space a template is provided by the user in order to allow the tuple to be found. For example, the template [?int,?boolean,"Kavi"] is one of many templates which matches the example tuple. Here we use the notation ?type to indicate the type of the field at that position. Therefore, this template matches any tuple which has an integer as its first field, a Boolean value as its second field and the string "Kavi" as its third field.

The exact syntax of tuples, templates and primitives depends largely on the syntax of the host language. The Linda model provide some basic operations to manipulate tuples stored in tuple spaces:

out TSm,tuple): Stores the tuple in the tuple space TSm.
in (TSm,template) This attempts to match the tuple template with a tuple in the tuple space and return it. If a match does not exist the operation blocks until a suitable tuple becomes available (if ever). If the match succeeds, the matched tuple is removed from the TSm.
rd (TSm,template) Same as in but non-destructive, that is, the tuple is copied as opposed to removed.
eval (TSm,tuple) This creates so-called active tuples each element of the tuple is evaluated concurrently and, when complete, the resulting tuple of values is placed in a tuple space. This is Linda's mechanism for spawning new processes.
handle = tsc(): Creates a new local tuple space and assigns a unique identifier to handle.

The Linda model is intended to be an abstraction, and as such is independent of any specific machine architecture. This has meant that alternatives and extensions to the basic Linda model have been proposed and investigated. The extensions that are currently supported in the York kernel are:

4.1 Multiple tuple spaces

The various ways in which multiple tuple spaces may be added has been discussed for some time. Currently, York has adopted the idea that each tuple space is independent of any other tuple space. There are however other ways of implementing multiple tuple spaces, such as using a hierarchical structure. The addition of multiple tuple spaces is achieved by incorporating a tuplespace type and a primitive to create a new tuple space within the model.

4.2 Collect primitive

A new tuple space primitive[1], collect(), has been proposed which subsumes the inp and rdp primitives. Given two tuple space handles (ts1 and ts2) and a tuple template, the primitive collect (ts1, ts2, template) moves tuples that match the template in ts1 to ts2, returning a count of the number of tuples transferred.

4.3 Copy-collect primitive

This is another new tuple-space primitive, which has recently been proposed in the light of work on the implementation of many different algorithms in Linda. This primitive is very

similar to collect but it is non-destructive. Hence it copies all the tuples that match the tuple template to a separate tuple space rather than moving them. As with collect it returns a count of the number of tuples copied.

5. LINDA'S OPERATIONS

In a distributed environment, processes have to work jointly with each other in order to solve a problem. The communication mechanisms are hard due to dependencies between the components. Therefore Coordination deals with managing dependencies between activities, a Coordination language is a good candidate to deal with the complexities that naturally arise in distributed environments[10]. The practical use of Linda can be seen in recent implementations: JavaSpaces by Sun Microsystems and TSpaces[9] by IBM, which were developed under the strong coordination of Linda. TSpaces was developed with a notion of persistence and it can perform many duties of a simple database system as it has indexing and query capabilities similar to a relational database. JavaSpaces offers transactions to the users to maintain data integrity[7].

5.1 Representation of Tuples

The general approach to build tuples is to use functions with a variable number of parameters, where each parameter represents a field. Having a subprogram with variable number of parameters is in some languages acceptable, but in linda the number of formal parameters in a subprogram must be fixed. In Linda the same effect is achieved by having default values for the parameters.

Procedure TS_Out (F0, F1, F2 : in Field := Empty Field);

In elinda the fields above must be of the same type, implying a alternative record or a pointer to a variation record, or there must be as many versions of TS_Out as there are combinations of fields. Because of it is possible to solve this problem more stylishly with tagged types. Tuples can be represented as a type instead of having procedures constructing tuples from a number of field parameters. Since a tuple is an ordered set of values it makes sense to map this onto a record, where a field in a tuple corresponds to a field in the record.

type Some_Tuple is abstract tagged record ... end record;

5.2 Representation of Actual and Formal Fields

The information whether a field is a formal or an actual field is indicated separate from the template. Unfortunately this cancels some of the benefits of having a tuple type. The tuple type ensures that fields can't mistakenly be transposed or forgotten, but it is still possible to transpose actual and formal indicators. Therefore , once the mode informationis given it is stored in an array of details literals in the template and there is no further risk for modes being transposed or forgotten.

5.3 Type Safe Tuples

The type profile of a tuple is called its signature. Only tuples with the same signature can maybe match. This is like comparing an integer with a float, they can never be equal since they are of different types. On the other hand two integers might match if they have the same value. In our implementation we extend the notion of signature slightly since we have typed tuples. This means that tuples having the same type profile can never match if they are of different types. As a side effect it is possible to use the tag of a tuple type to represent the signature. This use of the signature makes it possible to prevent tasks that are not intended to communicate from accessing each other's data by mistake.

5.4 New Operations on Tuple Space

Two new operations on TS are added, TS_Inp and TS_Readp. The operations doesn't belong to the original Linda model, but has been implemented in C-Linda. The operations try to find a matching tuple and return false if they fail, otherwise they return true and match with the found tuple. The only difference compared to TS_Inp and TS_Read is that the predicates will not block if no matching tuple is found. The new operations are very useful when the calling process can't afford to be blocked until a tuple arrives in the tuple space.

Example: - A server is capable of servicing many different requests. Each type of request is sent using a tuple of a different type and requests have different priorities. A request may only be serviced if no higher prioritized request is waiting. TS_Inp or TS_Readp can then be used to check for requests from high to low priority without blocking the server.

6. LINDA CHARACTERISTICS

Linda augments is the serial programming language (C or Fortran). It doesn't replace it, nor make it out dated. In this way, Linda builds on investments in existing programs.

Linda parallel programs are portable, and they run on a large variety of parallel computer systems, as well as, distributed memory computers, shared-memory computers, clusters, and networks of computers. With few exceptions, Linda programs written for one environment run without change on another.

Linda is easy to use. Conceptually, Linda implements parallelism using a logically global memory i.e. virtual shared memory, called tuple space, and a small number of simple but powerful operations on it. Tuple space and the operations that act on it are easy to understand and rapidly mastered. In addition, the C-Linda and Fortran-Linda compilers support all of the usual program development features, including compile-time error checking and runtime debugging and visualization.

7. CONCLUSIONS

This work presented an immediate messaging-based Coordination system for multi-agents by using Linda like coordination model. Our approach avoided spatially coupled condition and temporally coupled condition by using the linda technique for mail service. These two models meeting-oriented and blackboard-based Coordination model can also avoid spatially and temporally coupled conditions, but our approach is easier to implement other Coordination model. Not only agents but also real persons can coordinate with each other by using our system. This linda like coordination model is also used in parallel and distributed computing. This is easier to implement and it makes a user friendly interface for communication. In the future, the agent communication protocol could be extended in our system. Currently, all messages transferring via our system is text mode. The agent communication language (such as KQML [12]) could be combined with our system. Besides, although our system prototype is implemented with MSN messenger, other immediate messaging systems should be integrated to support a heterogeneous immediate messaging environment. This

model is widely used in newly application because of its strong characteristics.

8. REFERENCES

[1] A. Gerardo and Y. Yonco, "Communication Language and Protocols in an Agent-based Collaborative Learning Environment," Proceedings of the IEEE Systems, Vol. 3, pp. 2078-2083, October 1996.

[2] A. Gibaud and P. Thomin, "Communications Directed by Bound Types in Linda: Presentation and Formal Model," IEEE Transactions on Parallel and Distributed Systems, Vol. 13, NO.8, pp.828-843,August 2002.

[3] A. Nedelec, P. Reignier, and V. Rodin, "Collaborative Prototyping in Distributed Virtual Reality using an Agent Communication Language," Proceedings of the IEEE Systems,Vol.2,pp.1007-1012, 2000.

[4] D. Gelernter and N. Carriero. Coordination languages and their significance. Communications of the ACM, 35(2):97–107, 1992.

[5] D. Wang, N. Paciorek, and D. Moore, "Java-based Mobile Agents," Communications of the ACM, Vol. 42, No. 3, pp. 92- 102, March 1999.

[6] E. H. Durfee, "Scaling Up Agent Coordination Strategies," IEEE Transactions on Computer, Vol. 34, No. 7, pp. 39-46, July 2001.

[7] F. Zambonelli, G Cabri, and L. Leonardi, "Mobile-agent Coordination Models for Internet Applications," IEEE Damactions on Computer, Vol. 33, No. 2, pp. 82-89, Feb 2000.

[8] G. Cabri, L. Leonardi, and F. Zambonelli, "Engineering Mobile Agent Applications via Context-Dependent Coordination," IEEE Transactions on Software Engineering, Vol. 28, No. 11, pp. 1039-1055, November 2002.

[9] IBM Corporation. T Spaces Programmer's Guide, 1998. Electronic version only. http://www.almaden.ibm.com/cs/TSpaces/.

[10] "Mobile Agent Reactive Space,"http://polaris.ing.unimo.it/MOON/MARS/index.html.

[11] R. De Nicola, G L. Ferrari, and R. Pugliese, "KLAIM: A Kemel Language for Agents Interaction and Mobility," IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 315-330, May 1998.

[12] S . A. Moore, "KQML & FLBC: contrasting agent Communication languages," Proceedings of the 3hd Annual Hawaii International Conference. Vol. 6, pp. 6036, January 1999.

[13] "The Ara Platform for Mobile Agents," http://wwwagss.inforatik.uni kl.de/Projekte/Ara/index_e.html

[14] "The Home of the Mole," http://mole.infonnatik.uni-stuttgart.de.