INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

ANALYSIS AND DESIGN OF NSP: A NEGOTIATION SERVICE PROVIDER

Caroline Hakim

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

April 2000

Abstract

Analysis and Design of NSP: A Negotiation Service Provider

Caroline Hakim

Providing services is rapidly growing to be the trend of the future because of the added value it delivers to consumers. The growth in providing services is reflected in the development of new classes of service providers: Internet Services Providers (ISP) and Applications Services Providers (ASP).

In a heterogeneous dynamic community of software agents, that communicate with each other, conflicts are unavoidable. Negotiation among the conflicting agents is one way to resolve conflicts. Providing an 'independent service' to the community of agents that is specialized in handling negotiations, is the focus of this thesis. We propose a Negotiation Service Provider (NSP) that provides negotiation services to an agent community. When an agent detects a conflict, it contracts a NSP to negotiate on its behalf. NSP investigates the conflict and, selects the most suitable negotiation protocol available from its protocol suite. With this protocol, NSP conducts the negotiation with the conflicting agent on behalf of its client agent.

In this thesis, we design an architecture for a NSP. The different modules of this architecture and their interactions are presented. Then, we discuss the integration of NSP into the agents' community. We propose a representation scheme for conflicts between agents. This scheme along with a data structure called Protocol-Constraint Table (PCT) guides NSP's negotiation agent in the protocol selection process. We introduce a method to integrate new negotiation protocols in NSP's protocol suite. This method is demonstrated by using three classical negotiation protocols.

Acknowledgments

This thesis has seen the light of day thanks to the great understanding, valuable support and financial support of my supervisor Dr Radhakrishnan. He encouraged me to embark on my graduate studies and has given me the freedom to investigate areas which are off the beaten track which eventually lead to this work. If I have managed to complete this work, it is definitely because of the support, encouragement, pushing and challenge of many people in my life.

It is the unconditional love of my nieces and nephews, Christine, Georges, Marc, Eric, Alain, Stéphanie and Giorgia, that help me embrace a new day every morning. My mother has been more than a "real mom" as Paul says, and has taken extra care of me the past few months to help me go through the writing. I owe my studies to the financial sacrifice of all the members of my family. They believe in me and I hope I don't disappoint them too often. Without my sisters and my family, I wouldn't have been here today. Without my cousin and best friend, Rita, life would be unbearable.

Stan has offered to help me with LaTeXon the condition that I don't mention his name here. Stan, I lied. Stan did not just help me with formatting my thesis, but he also reminded me of my deadlines when I needed it the most. If it wasn't for my working in the analysts pool and learning so much from the masters, I would have never come up with my "masters". Michael's consultations over lunch time and in corridors affected this work. My study bodies, first Andrea and then Priya, have helped me get on track and stay there to complete the thesis. Kim has been wonderful in proof reading my writing. Space does not permit to thank all my friends who showed their care and support to bring to conclusion this chapter of my life: Paul, Galina, Monia, Spanner.

Finally, I must mention the support and understanding of Larry, François and Ghassan, which helped me work on my thesis while pursuing a career.

Contents

List of Figures

List of Tables

List of Acronyms

ACL:	Agent Communication Language
ACLS:	Agent Communication Language Suite
ANSI:	American National Standards Institute
ASP:	Application Service Provider
COM:	Component Object Model
CORBA:	Common Object Requester Broker Architecture
DARPA:	Defense Advanced Research Projects Agency
DCOM:	Distributed Component Object Model
DEC:	Digital Equipment Corporation
DF:	Directory Facilitator
DNA:	Digital Network Architecture
EGP:	Exterior Gateway Protocol
FDDI:	Fiber Distributed Data Interface
FP:	Feasibility Preconditions
FIPA:	Foundation of Intelligent Physical Agents
GIOP:	General Inter-ORB Protocol
HA:	Hotel Agent
HSSI:	High-Speed Serial Interface
HTML:	Hyper Terminal Markup Language
HTTP:	Hypertext Transmission Protocol
IGRP:	Interior Gateway Routing Protocol

IIOP:	Internet-ORB Protocol
IP:	Internet Protocol
IPC:	InterProcess Communication
ISDN:	Integrated Services Digital Network
ISP:	Internet Service Provider
IS-IS:	Intermediate System to Intermediate System
KAoS:	Knowledge-able Agent-oriented System
KIF:	Knowledge Interchange Format
KQML:	Knowledge Query and Manipulation Language
KSE:	Knowledge Sharing Effort
MAGNET:	Multi-Agent Negotiation Testbed
MAS:	Multi Agent Systems
MM:	Management Module
NA:	Negotiation Agent
NAG:	Negotiation Agent Generator
NSP:	Negotiation Service Provider
OA:	Ontology Agents
OMG:	Object Management Group
ORB:	Object Request Broker
OSI:	Open Systems Interconnection
OSPF:	Open Shortest Path First
PC:	Personal Computer

PCT:	Protocol-Constraint Table
PGG:	Partial Global Goals
PGP:	Partial Global Plan
PS:	Negotiation Protocol Suite
PTA:	Personal Travel Agent
RE:	Rational Effect
RIP:	Routing Information Protocol
RMI:	Remote Method Invocation
RPC:	Remote Procedure Calls
SDLC:	Synchronous Data Link Control
SL:	Semantic Language
SMDS:	Switched Multimegabit Data Service
TA:	Travel Agent
TCP:	Transport Control Protocols
TCP/IP:	Transport Control Protocol / Internet Protocol
UDP:	User Datagram Protocol
VINES:	Banyan Virtual Integrated Network Service
WWW:	World Wide Web
XNS:	Xerox Network System

Chapter 1

Introduction

The world is rapidly moving in the direction of providing newer types of services. The services sector has a major impact on the growth of the economy of major countries such as Canada and the United States. "The services sector continues to be the growth engine of the U.S. economy, accounting for 73% of all American jobs in 1998, up from 72% in 1997." [url99a] The rapid advances of computing technology highly contribute to the growth of the services sector. "However, some countries are realizing greater benefits from improved service sector performance than others. In these countries, the development of knowledge-based services, often linked to information technology, appears to be an important source of growth." [url] The greater potential in providing services is reflected in the development of new classes of service providers: Internet Services Providers (ISP) and Applications Services Providers (ASP).

In this thesis, we introduce yet another service provider, called NSP. We propose to provide negotiation services through this Negotiation Services Provider (NSP). Negotiation skills are crucial for communication. Therefore, NSP is of particular interest in the context of a society of agents where agents are continuously communicating amongst each other. We put forward the requirements specification of NSP.

In the remaining chapters, we will study NSP from an analysis and design perspective. We will investigate the research fields that form the basis of NSP. These fields are: providing services, multi-agents and negotiation protocols.

1.1 Providing Services

Providing services is increasingly shaping to be the trend of the future mainly because of the added value it delivers to consumers. The service providers have considerable potential to turn their services into lucrative businesses. Consider the following quote: "At Sun, we believe that service providers are the cornerstone of the new Net economy - and we intend to accelerate their success." [urlf] Thus, Sun Microsystems is designing its hardware and servers to meet the needs of service providers companies. It is even going one step further and has established the SunTone Certified Program as a standard for service quality [urlg].

Internet Service Providing, one of the first services available electronically, is a faithful illustration of the success of this field. From the business perspective, ISP companies such as America Online and Prodigy, have grown so large that main stream computing companies such as IBM and Microsoft have moved in that direction and are now providing Internet services too. The potentials of this field is extended from business to households in a pervasive way.

The current major success is ASP and ASP companies have already made considerable benefits. Companies such as Oracle, Nortel, Microsoft, and Siebel have adopted this new technology.

1.2 Internet Service Providers (ISP)

Internet Service Providers (ISP) are companies that provide users with a connection to access the vast computer networks of the Internet. Usually, ISPs offer services such as: Web hosting - Domain Name Services - Proprietary Online Services. They supply their customers with a software package, a username and a password. Typically, users would phone the ISP servers to log on and access the Internet but this technology has evolved and dial up connections over regular modems have been replaced by cable modems, and Integrated Services Digital Network (ISDN). With their network access, users are able to browse the World Wide Web (WWW) and USENET, as well as send and receive e-mail [urlh].

ISPs have been the principal gateway for households to the Internet. In fact, the package of services provided by ISPs has attracted not only individuals but also large companies. Instead of building a network infrastructure from scratch, connecting it

to the Internet and growing in-house expertise in managing it, many companies opted to benefit from these facilities and tools by using the services of third parties namely ISPs for a monthly fee. A company will use specialized ISPs for the following reasons:

- Fast setup: Building a network and connecting a network takes much longer than just installing a modem and the accompanying software package in the computer to dial in to the ISP.
- Cheap maintenance: Hiring a technical team to build the infrastructure and maintain becomes a major overhead if the company does not extensively use the Internet services.
- Simple setup: Installing a modem and its software to dial out is simpler and less prone to errors than setting up an entire network with direct access to the Internet.

1.3 Application Service Providers (ASP)

One of the ASP companies is USinternetworking. Six months after going public, with annual revenues of less than $30 million, USinternetworking is already capitalized at $2.6 billion [urli]. Sun also quotes that IDC, a technology research firm, expects that ASP-related spending worldwide, at an estimated $150.4 million in 1999, will grow and exceed the $2 billion by 2003 [urli]. Less conservative estimates are put forward by Mindbridge in a press release. Mindbridge announces that the ASP market is estimated to hit $2 billion in 1999 and predicts a compounded annual growth rate of 40% over the next three years [url99b]. As defined by AOL pcwebopedia: "Application Service Providers are third-party entities that manage and distribute software-based services and solutions to customers across a wide area network from a central data center. In essence, ASPs are a way for companies to outsource some or almost all aspects of their information technology needs." [urlj]

ASPs responsibilities range from simply hosting to fully installing and managing a wide range of business applications on behalf of their customers. Therefore, a client is able to add new applications to the subset it is already renting transparently. The best ASPs securely deliver a variety of applications over a centralized, high performance

infrastructure to geographically distributed customers. They also have specialized support to help their customers in their transitions.

The following are the major benefits of ASPs [url99b, urli]:

- By relying on ASPs, companies can concentrate on their line of business to build their competencies and competitive advantage.
- ASPs eliminate the operational overhead of their customers.
- While guarantying a complete solution, ASPs minimize deployment time and cost.
- ASPs offer on-the-fly, unlimited scalability that many small IT departments lack at a reasonable cost.
- ASPs would typically have experts IT staff specialized in installing and managing applications.

1.4 Negotiation Service Providers (NSP)

As an outcome of the research in the field of artificial intelligence we now have what is termed "intelligent software agents". Societies of agents are formed where multiple agents communicate among each others. We will not go into the debate of defining what intelligence is or what an intelligent agent is. In fact, the Artificial Intelligence (AI) community has not reached a consensus in defining these concepts. For the purpose of our study, an intelligent agent is an entity built to accomplish a specific task or to pursue a specific goal. It is able to:

- Communicate with other agents
- Make decisions
- Alter its actions

These agents may potentially use the Internet as a communication medium. Wherever communication is available, conflicts are an immediate threat. Therefore, negotiation between agents is an essential aspect in multi-agent systems. In their hurry in bringing an agent to life, developers seem to underplay or ignore the negotiation

aspects of their agents. In the process of introducing negotiation abilities to their agents, developers are confronted with two major setbacks.

The first limitation is at the level of the variety of negotiation protocols incorporated in the multi-agent system. Designing and implementing an agent to solve a specific problem is quite a complex task. In the light of all the requirements of an agent based software development, software engineers rather disregard the negotiation aspects of their agents even when negotiation is critical for the application's domain. Acquiring and maintaining advanced negotiation skills is a long and difficult process. So when developers attempt to introduce some negotiation protocols into their multi-agent system, they implement one protocol. These protocols are traditionally hard coded in the agent.

The maintenance and update of the protocols implemented in the agent constitutes the second major difficulty. Even the developers who addressed the issues of negotiation will find their agent, at one point, limited to the already defined algorithms. The agent will not have the resources required to adopt the protocol adequate to the current circumstances. Furthermore, in order to incorporate new protocols, the agent will have to be rewritten. Developers prefer to concentrate on solving application specific problems rather than spend their scarce resources by embedding a selection of protocols in their agent and then struggling to maintain them. Further, the reusability and the sharing of these skills is restrained by both the environment of the application itself and by its owners and developers.

Providing independent services specialized in handling negotiations to the entire community of agents becomes a need rather than a luxury. We propose the development of a Negotiation Service Provider (NSP) which will offer its expert services to the agents community. From a design perspective, good design practice recommends grouping similar functionalities and their corresponding data within the same application into modules. The advantages of such an approach are: easier debugging, reusability of modules in other applications and simplification of maintenance among others. This is the spirit of Object Oriented design. NSP is based on the concept of providing services and thus shares the same approach as ISP and ASP in solving different flavors - Internet, applications, negotiation - of the same need for tools. The advantages of centralizing negotiation and providing it as a service to different clients are, in general, the same as the ones for ISP and ASP, most importantly:

- The degree of expertise in negotiation becomes a choice rather than a necessity.
- Developers may concentrate on the original problem rather than getting sidetracked with negotiation issues.
- Developers delegate the burden of keeping up with the latest developments in this domain to the specialized services.

1.5 Requirements Specification of NSP

Isolating the main stream of the application from the negotiation aspect is a major step towards a viable solution for NSP. NSP must have provision to overcome the limitations and constraints of its potential customers. The major requirements of a NSP are as follows:

1. Minimize the constraints on the agents.
2. Go beyond the language barrier in providing the services.
3. Provide a wide variety of protocols.
4. Be able to assess the conflict and adopt the most appropriate protocol for the situation.
5. Support the extension of the protocol repository.
6. Keep track of the rationale behind the decisions adopted by NSP and provide means to allow the client agent to retrieve the logs of negotiation as well as the rationale for reaching these decisions.
7. Support a dynamic network of agents.

Based on these requirements, NSP is designed with two dynamic repositories. One is formed of negotiation algorithms and the other one is composed of agent communication languages. These repositories are dynamic in the sense that they are able to grow and result in new algorithms or new languages. No specific ACL is imposed on the clients since NSP is "multi-lingual". NSP operates within the requirements and the level of authority set by its client before launching the negotiation process.

Depending on the conflict, NSP selects the most appropriate protocol to use in such circumstances. In the process of negotiating an agreement, all the available alternatives and the options considered along with the rationale followed in the decision process are logged.

1.6 Claims and Contributions

The Internet is a dynamic and heterogeneous environment where agents of different kinds will coexist. Their communication language will differ. When they negotiate with each other, their negotiation protocols could differ. Furthermore, there is a need to add new languages and protocols as the field matures. The NSP proposed in this thesis is a negotiation support system that is intended to facilitate the above dynamic and open environment. The heart of the claim is in the manner in which these changes will be accommodated now and in the future.

The major contribution of this thesis is in the innovative approach of addressing negotiation in a multi-agent environment. The current research in the field of negotiation is done at the micro level. It is concentrated on developing new algorithms to negotiate for different scenarios and on enhancing old ones. This proposal research looks at negotiation at a macro level. It builds an infrastructure to group the algorithms. In this environment, agents benefit from all the algorithms rather than just a subset. Further, scalability, maintenance and management issues are incorporated into the core design of NSP.

The merits of this thesis are not confined to a technical level, the business potentials of the proposed system are considerable. Because of the difficulty in developing negotiation skills, NSP could reduce the development time.

1.7 Thesis Outline

The thesis is divided into six chapters and five chapters follow this chapter:

Chapter 2 provides an overview of multi-agents. We examine the environment of multi-agents. We review the communication protocols and the various distributed computing standards. We also discuss communication issues among agents, specifically agent communication languages and ontologies.

In Chapter 3, we develop the design of NSP. We present our model for NSP along with its different modules. We illustrate the interactions among the various modules of NSP. Then, we describe the procedure to establish the initial contact between the agents and NSP. We conclude this chapter with an event trace of the different stages of providing negotiation services with NSP.

In Chapter 4, we closely examine conflicts. We define the context and the cause of the conflict from NSP's perspective. We introduce some additional tools with a direct impact on the results of NSP. The agent's own prioritized list of goals, the constraints inherent to the conflict with their margin of maneuver, and additional information of agents are the different inputs that play a crucial role in directing the negotiation of NSP.

We begin Chapter 5 with a description of three classical negotiation protocols. We introduce our method to both integrate negotiation protocols in NSP and match protocols to conflicts in terms of their constraints. Then we illustrate this method by classifying the protocol presented earlier according to the protocol-constraint table.

Chapter 6 is the conclusion of this thesis. We summarize the achievements reached through the analysis and design of NSP. We conclude the chapter with a list of future research and open issues.

Chapter 2

Overview of Multi Agent Systems

In their article "A Perspective on Software Agents Research" [NN99], Nwana & Ndumu (1999) define multi-agent systems and state: "The hypothesis/goal of multi-agent systems (MAS) is clear enough ... creating a system that interconnects separately developed agents, thus enabling the ensemble to function beyond the capabilities of any singular agent in the set-up." In a dynamic environment where agents are created and destroyed, communication raises a few challenges. What protocols are suited to interconnect agents running on separate hosts? How are the agents to be distributed and coordinated? Another challenge in MAS is at the level of the agent communication language. A language is used for message communication. The ontology or "concept definitions" is a major issue in a common language for the MAS. Agreeing and sharing a common definition for the concepts are a prerequisite for effective communication.

2.1 Heterogeneous and Dynamic Environment

Nwana and Ndumu's description of a MAS as "separately developed agents" [NN99] introduces two implicit characteristics proper to the environment of multi-agent systems. It is both dynamic and heterogeneous.

Developing agents separately introduces a time factor. A multi-agent system is not necessarily built at one time; it would evolve. During its lifetime, new agents are introduced in the MAS either to replace older agents or to add new functionality to the entire system. The continuous potential for changes in the life cycle of the MAS

elements makes its environment highly dynamic.

In a MAS, there is no restriction on the sources of the agents being interconnected. Hence, two agents developed by two distinct sources may potentially be either interlinked together or incorporated to an even larger MAS. Differences at the level of the developers of the various agents forming the MAS illustrates one aspect of the heterogeneity of the environment. Even within a single development platform for the entire systems, differences among the agents might arise to meet the purpose of the MAS. This heterogeneity entails the following potential differences:

- At the design level of the agents themselves
- At the level of the agent communication language used by each agent.
- At the level of the implementation of the agents.
- At the level of the functional specialization of each agent.

These two characteristics of heterogeneity and dynamic nature of MAS have a deep impact on the different aspects of a multi-agent system discussed in the rest of this chapter.

2.2 Communication Protocols

"In the context of data networking, a protocol is a formal set of rules and conventions that governs how computers exchange information over a network medium." [urla] As depicted in Figure 1, Communication protocols are divided into the following types: LAN protocols, WAN protocols, routing protocols and network protocols. LAN protocols operate between the physical layer and the data link layer of the Open Systems Interconnection (OSI) Reference model. Examples of LAN technology are: Ethernet invented by Xerox, Token Ring network developed by IBM, and Fiber Distributed Data Interface (FDDI) developed by the American National Standards Institute (ANSI). WAN protocols operate at the lowest three layers of the OSI model: the physical layer, the data link layer, and the network layer. Examples of WAN protocols include Frame Relay, High-Speed Serial Interface (HSSI), Switched Multimegabit Data Service (SMDS), Synchronous Data Link Control (SDLC), X.25. Routing protocols are the implementation of routing algorithms. They direct protocols through

the network. Interior Gateway Routing Protocol (IGRP), Enhanced Interior Gateway Routing Protocol (Enhanced IGRP), Open Shortest Path First (OSPF), Exterior Gateway Protocol (EGP), Border Gateway Protocol (BGP), Intermediate System to Intermediate System (IS-IS), and Routing Information Protocol (RIP) are all examples of routing protocols. Network protocols also known as routed protocols function at the upper four layers of the OSI model: the transport layer, the session layer, the presentation layer, and the application layer. Different protocol suites address different functions. Example of network protocols are: Internet Protocol (IP), DECnet, AppleTalk, Novell NetWare, Banyan VINES, and Xerox Network System (XNS). Operating at the upper layers of the OSI model, network protocols are the closest to the software applications. Because network protocols interact directly with our application namely NSP, we will investigate these protocols in further detail. For an elaborate study of the communication protocols we refer to the book "Internetworking Technology Overview" which is part of Cisco's Documentation CD-ROM available online [urlk].

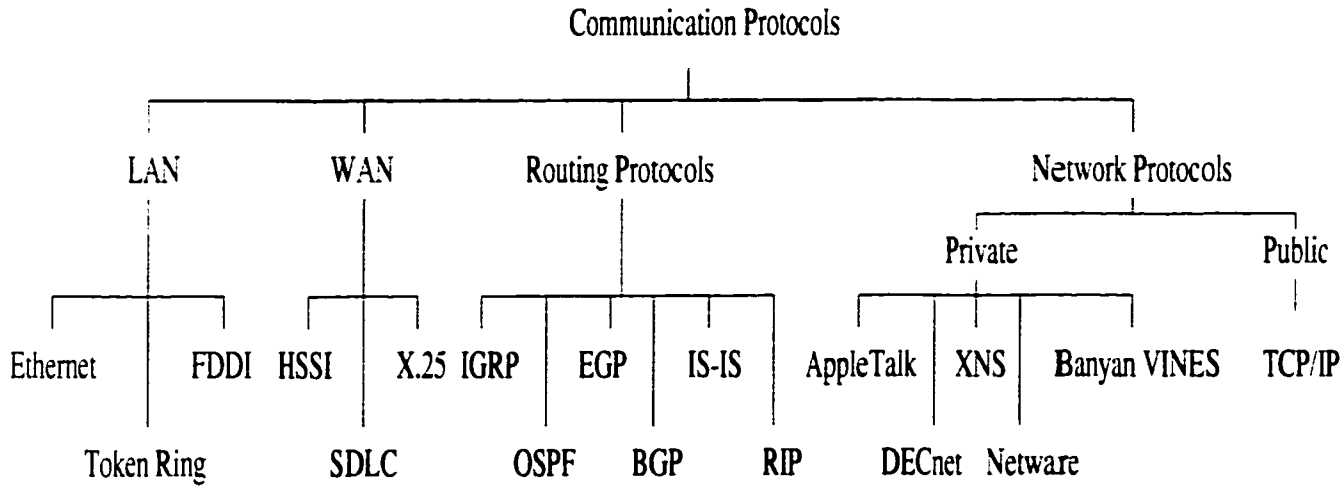

Figure 1: Communication Protocols Taxonomy

Two major models divide network protocols: the private model or the public model. In the private model, equipment is homogeneous to a large extent and the network protocols are highly proprietary. Typically, these protocols are supported by specific vendors on specific architectures. AppleTalk is a protocol suite developed by Apple Computer in conjunction with Macintosh computer in the early 1980s. Although AppleTalk is supported on PCs too, it's usage is highly limited to networks

with a large percentage of Macs. The protocol suite included in DECnet is developed and supported by Digital Equipment Corporation (DEC). DECnet is based on Digital Network Architecture (DNA). It mainly connected VAX minicomputers. Only recently, Digital supported non proprietary protocols. Based on Xerox Network Systems (XNS), NetWare was developed by Novell, Inc. It specifies the upper five layers of OSI and runs on various computer architectures from PCs to mainframes. However, NetWare protocol suite is also based on proprietary protocols. Banyan Virtual Integrated Network Service (VINES) is also based on a proprietary protocol family derived from Xerox Network Systems (XNS) protocols. Xerox Network Systems (XNS) protocols were developed by Xerox Corporation but their derivatives in PC networking implementations such as NetWare and Banyan VINES gained more popularity. Several XNS protocols resemble the IP and Transport Control Protocol (TCP) protocols. Thus, in the private model, proprietary confined islands of computers were created with no bridges to connect them.

The public model includes all open-system protocols functional in any set of interconnected networks. The world's most popular example of such network protocols is the Internet Protocols suite funded by Defense Advanced Research Projects Agency (DARPA). It has been developed with the goal of heterogeneous connectivity in mind. Open and free, this suite had a considerable edge and resulted in it being at the foundation of the Internet and the World Wide Web (WWW).

Because NSP targets a heterogeneous environment, the Internet Protocols suite also known as TCP/IP would be the most appropriate protocol.

2.3 Distributed Computing Standards

Several standards with different levels of sophistication have emerged to support distributed computing. Object Management Group's (OMG) Common Object Requester Broker Architecture (CORBA), Microsoft's Component Object Model (COM) and Distributed Component Object Model (DCOM), Sun Microsystems' Remote Method Invocation (RMI) are the three major competing standards which support object-oriented distributed computing by allowing remote invocation of object methods. These models are based on the object oriented paradigm. Message based InterProcess Communication (IPC) and Remote Procedure Calls (RPC) are commonly

used in multiprocessing systems. IPCs and RPCs are similar protocols allowing processes rather than objects to exchange information among each other or execute programs on other computers.

CORBA is an industry standard for connecting distributed programs running on different operating systems and written in different languages over the Internet without any special requirements other than knowing the services available and the name. CORBA provides an interface called General Inter-ORB Protocol (GIOP) for writing programs. GIOP uses TCP/IP as its transport protocol. "The mapping of GIOP message transfer to TCP/IP connections is called the Internet-ORB Protocol (IIOP)" [urlb] IIOP is a protocol which permits the exchange of integers and various complex objects between servers and browsers [urlc]. With this feature, IIOP expands HTML which is limited to the transmission of text. IIOP is an integral part of CORBA as depicted in the Figure below.

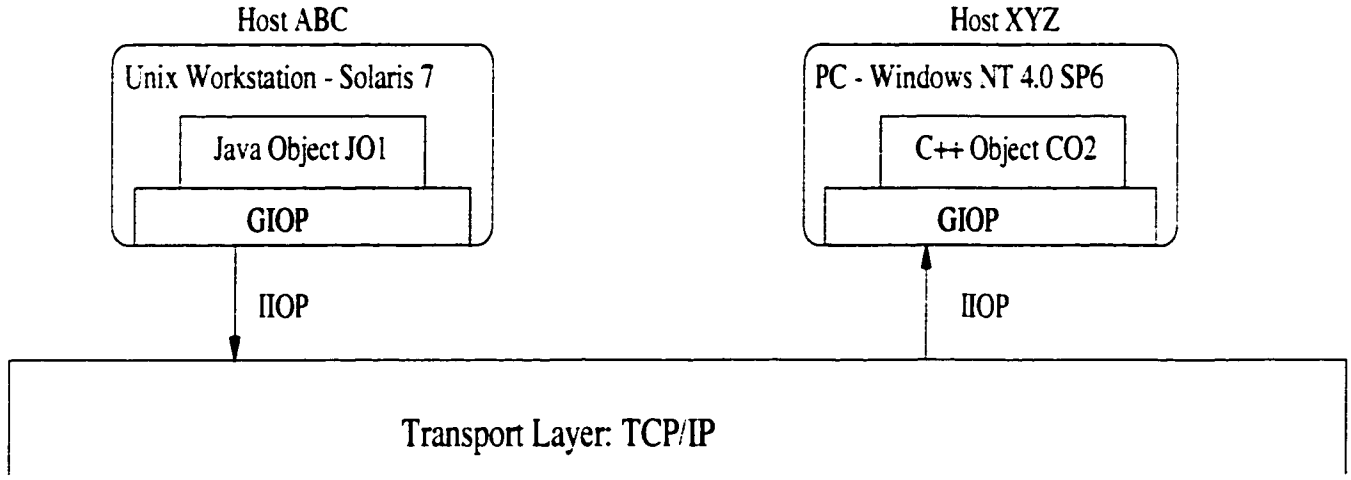

Figure 2: OMG's CORBA

In Figure 2, hosts A and X are two computers different both at the hardware level and at the software level. A is a Sun Ultra workstation running Unix Solaris 7. A Java client application is running on A and has a Java object called JO1. X is a PC running Microsoft Windows NT 4.0. The server running on X is implemented in C++ and it has an object called CO2. To invoke the method provided by CO2 running on X and which is using a different ORB, JO1's request goes through GIOP, a standard wire protocol. Because A and X are connected through the Internet, CORBA's IIOP translates GIOP messages into TCP/IP at A and the TCP/IP messages back into

GIOP at X's end. The common requirements for the two hosts, GIOP, is shown shaded in Figure 2.

While CORBA is supported on various operating systems, Microsoft's DCOM is currently restricted to Windows. COM and DCOM are only available on Windows 95/98 and NT platforms. Nevertheless, no restrictions are put at the language level. A DCOM object may be written in any language and it is able to communicate with other DCOM objects regardless of which language is used in implementing them. DCOM uses TCP/IP and HTTP. HTTP is an even higher application protocol which is based on the TCP/IP suite of protocols to get to the Internet.

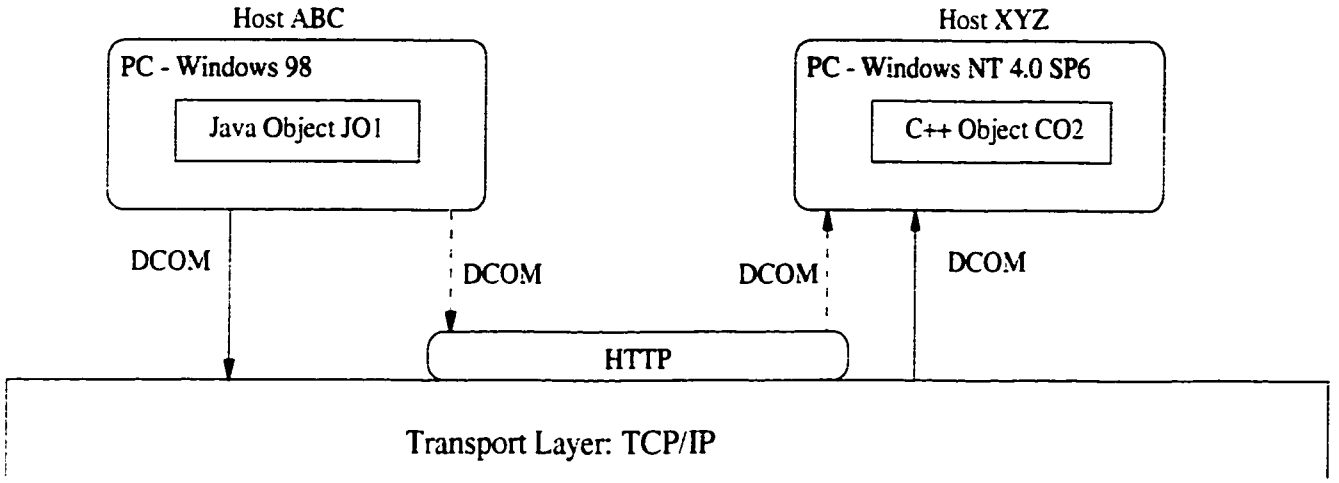

Figure 3: Microsoft's DCOM

Figure 3 shows two hosts A and X using Microsoft's DCOM protocol. Both hosts may have quite dissimilar hardware but only to the extent where Microsoft operating systems support it. The restriction on the operating system is shaded in Figure 3. A is running Windows 98 whereas X is running Windows NT 4.0 with Service Pack 6 applied. The Java Object JO1 belongs to the Java client running on A. JO1 is implemented in Java. X is running a server implemented in C++. One of its objects is CO2. Using the DCOM protocol, JO1 invokes a method of CO2. JO1's request is communicated to CO2 over TCP/IP. This is done by either directly using TCP/IP or by going through HTTP.

Both CORBA and DCOM impose no restrictions on the programming language and are designed to allow objects to communicate with each other regardless what programming language they were written in. However, Sun Microsystems' RMI is a

relatively simple set of protocols, developed by Sun's JavaSoft division, which only works with Java objects. Within RMI, Java objects can communicate with other Java objects only. However, because Java objects are not restricted to specific architectures or operating systems, RMI aren't either. The RMI transport layer uses TCP/IP by default. However, RMI socket factories allow the use of a non-TCP or custom transport layer over IP [urld].

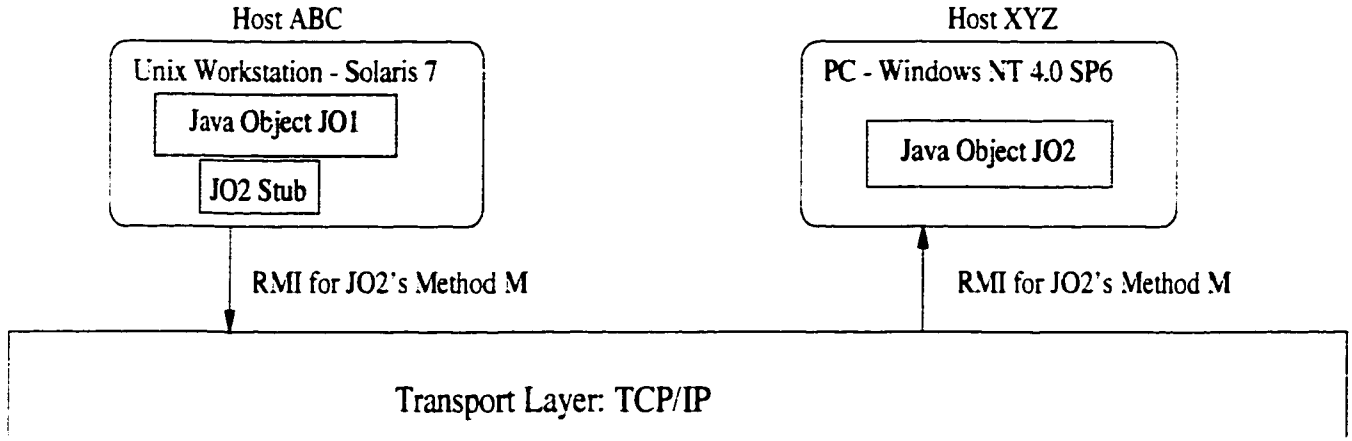

Figure 4: Sun Microsystems' RMI

In Figure 4, host A is a Unix workstation running Solaris 7 whereas X is a PC running Windows NT 4.0 with Service Pack 6 applied. Both hosts are running Java virtual machines and both the client and the server are implemented in Java. As the common requirement for the two interconnecting hosts, Java is shaded in Figure 4. Host X is running the server application which created JO2. The server provided references to JO2 and thus made it possible to invoke the remote object JO2 across virtual machines by providing references to it. The client application includes a Java Object JO1. The client running on A receives a reference to JO2 and invokes JO2's method M. The communication between the server and the client is done through RMI.

By minimizing the constraints in connecting the distributed applications, OMG's CORBA is able to support the most heterogeneous systems. Since heterogeneity is at the core of MAS, CORBA is recommended to handle the data transfer and the communication among agents in NSP.

2.4 Agent Communication Languages

Agents need a common language to communicate among each other. The Foundation for Intelligent Physical Agents (FIPA) [url97] defines Agent Communication Language (ACL) as "a language with precisely defined syntax, semantics and pragmatics that it the basis of communication between independently designed and developed software agents". "An ACL provides agents with a means of exchanging information and knowledge" [YLP99]. The communication among agents is not limited to a single message. On the contrary, it is a conversation based on task dependent sequences of messages. Although some agent systems applications such as KAoS [ea97] use CORBA's architecture solely, ACLs are not alternatives to the distributed computing mechanisms mentioned in the previous section. In fact, ACL messages may often be delivered via such mechanisms. Labrou, et al. observe that ACLs form one level above CORBA for two reasons:

- ACL messages describe propositions, rules and actions rather than objects with no semantics. This is unlike CORBA level message communication.
- ACL messages express a desired state in a declarative language instead of a procedure or method as it is the case with CORBA.

ACLs emerged within the Knowledge Sharing Effort (KSE) initiated by the Defense Advanced Research Projects Agency (DARPA) of the US Department of Defense. It investigated knowledge sharing and reuse by developing techniques, methodologies and software tools at design, implementation and execution time. "The central concept of the KSE was that knowledge sharing requires communication, which in turn requires a common language" [YLP99]. The KSE model is a three layer model as shown in Figure 5. These layers are independent from each other.

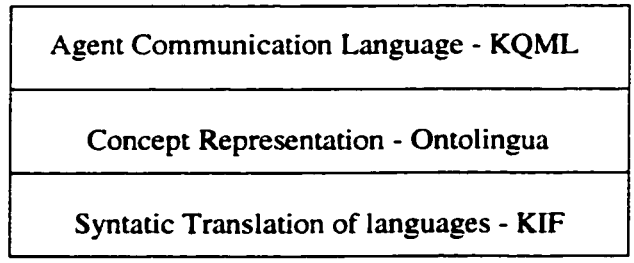

Figure 5: KSE Model

The bottom layer is concerned with the problem of heterogeneity of knowledge representation at the level of syntactic translation between variations of the same language or different families of languages. KSE proposed Knowledge Interchange Format (KIF), a logic language "as a standard for describing things within computer systems such as expert systems, databases, intelligent agents, and so on." [YLP99]. Further, KIF was designed as an interlingua, an intermediary language in the translation of other languages.

The middle layer addresses the ontology problem. When the same concept is defined differently in two applications communicating with each other, a common ontology is needed. Labrou et al define an ontology as "a particular conceptualization of a set of objects, concepts, and other entities about which knowledge is expressed, and of the relationship among them." [YLP99] Ontolingua is KSE's solution. We will discuss the ontology problem further in the next section.

The top layer is the communication language among agents. This communication is at the level of desired states. KSE's ACL is Knowledge Query and Manipulation Language (KQML). KQML is an inter-agent, high-level and message-oriented communication language and protocol. Conceptually, KQML has a three layer organization shown in Figure 6: the content layer, the message layer and the communication layer.

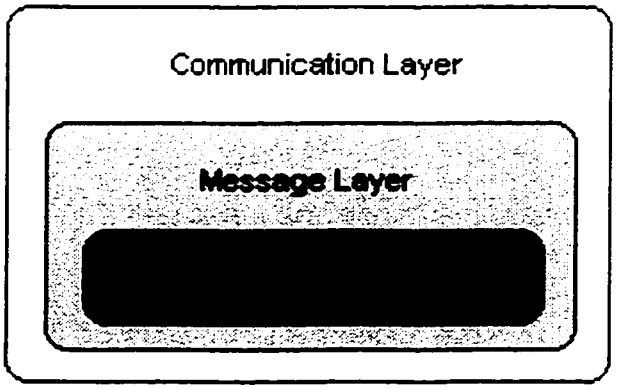

Figure 6: KQML Model

The content language carries the actual content of the message written in the program's own representation layer. The content of the message is wrapped inside a KQML message. Messages are opaque. KQML-agents are only concerned with the boundaries of the message and its delivery.

The message layer forms the core of the KQML message. It is used to encode the messages to be exchanged. It determines the possible kinds of interaction with an agent that understands KQML. It identifies the network protocol used in delivering messages and interprets a "speech act or performative", such as an assertion, a query or a command attached to the content by the sender. This layer includes optional features that could describe the content better. Such features simplify the implementation of the tasks of analyzing, routing, and properly delivering messages despite their opaqueness.

The communication level encodes the lower level communication parameters of the message such as the identity of the sender and receiver, a unique identifier for that particular communication.

The balanced-parenthesis list in the syntax of KQML is inherited from its initial implementations in Common Lisp. The first element of the list is the performative while the rest of the elements are its arguments in the form of keyword/value pairs. The following is a simple example of a message representing a basic query performative in KQML:

```
(ask-one
   :content (VACANCY TWA ?flight)
   :receiver american-express
   :language standard-prolog
   :ontology TRAVELING)
```

In this message, the performative is *ask-one*, the content is *(VACANCY TWA ?flight)*, the receiver of the message is *american-express*, the content of the message, i.e. the query, is written in *standard-prolog*, the ontology assumed by the query is identified by the token *TRAVELING*.

In the case of KQML certain primitives have pre-defined meaning and there are a couple of dozen of them. Implementations of those reserved performatives must comply with their original definitions in KQML. The set of KQML performatives is extensible. Additions to the set of performatives in a community of agents is achieved when all the agents agree on the definitions — interpretations and associated protocols — of the new performatives. KQML also introduces a special class of agents called "facilitators" to coordinate the interactions of other agents. Facilitators provide network and communication services such as maintaining a registry of service names,

delivering messages that are incompletely addressed, and finding appropriate clients and servers.

Originally, KQML only had an informal and partial semantics descriptions. In an effort to fill this incompleteness, Labrou and Finin provided KQML's semantics in terms of "conditions" [YLP99, Lab96, LF97, LF98]. Preconditions define the necessary state for an agent to send a performative without guarantying its successful execution and performance. Postconditions indicate the states of the sender and receiver assuming that the performative was successfully received and processed. Completion conditions of a performative describe the final state. The conditions are expressed in terms of mental attitudes (belief, desire, intention, knowledge) and action descriptors (for sending and processing messages). Although the language describing mental states restricts the combination of mental states, no semantic models for the mental attitudes are defined. The other semantic model based on earlier works qualifies KQML primitives as attempts at communication rather than performatives. Labrou et at. claim that the approach of this semantic "strongly links the ACL semantics to the agent theory assumed for the agents involved in an ACL exchange" [YLP99].

Currently, the alternative ACL to KQML is FIPA ACL developed within the Foundation of Intelligent Physical Agents (FIPA) organization. It is a non-profit organization. "FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment" by producing internationally agreed specifications to maximize interoperability across agent-based applications. These standards are established through an open international collaboration of member organizations both universities and companies, active in the field of agent hood [url97].

FIPA's community has been actively working on the standardization of various agent-based issues. It has been releasing several specifications over the past few years. One of the major specifications is the Agent Communication Language. The Agent Communication Language specifications provides a normative description of a set of communicative acts supported within FIPA ACL in terms of their pragmatics and their semantics. It also includes the normative description of a set of interaction protocols such as contract net and different kinds of auction.

Similar to KQML, FIPA ACL is based on speech act theory. An agent communicates its intentions to other agents using a special class of actions called communicative acts. Syntactically, FIPA ACL is identical to KQML except for name differences

of some reserved primitives. FIPA ACL messages are also opaque and delivered over a simple byte stream. No specification for the content of messages is provided by FIPA. The following message is an illustrative example in FIPA ACL.

```
(request
   :sender i
   :receiver j
   :content (action j (deliver box017 (location 12 19)))
   :protocol fipa-request
   :reply-with orders567)
```

The communicative act in this message is *request*, the sender is *i*, the receiver is *j*, the content of the message, i.e. the description of the action to be performed is *(action j (deliver box017 (location 12 19)))*, the protocol or the pattern of message exchange is *fipa-request* and the reply-with expression to use in the reply to this message is *orders567*.

Because no requirements are enforced on the syntax of the messages, communicating agents have the responsibility of ensuring that the messages expressed in ACL are mutually comprehensive. This may be achieved by following certain conventions, by negotiation or by using the translation services of a third party. However, for the purpose of key agent management functions, the ACL specification defines a content language which is an s-expression type notation. FIPA compliant agents are required to use this agent management content language and an ontology when carrying standard management duties. FIPA specification provides Semantic Language (SL) content language on an informative basis. Languages based on sub-grammars of the s-expression grammar produce legal ACL messages and do not need modification. If any other notations are used to encode messages, ACL has two techniques to extract the message without requiring ACL parsers to parse any expression in any language: wrap the expressions in double quotes making the message a string in ACL or prefix the expressions with the length of the string. Although the content of the message may be encoded in any language specified in the ":language" parameter, it must be able to express propositions, objects and actions. Propositions define the truth value of a sentence. Objects represent an identifiable abstract or concrete entity in the domain of discourse. Actions are constructs representing activities performed by some agent.

FIPA defines a set of standard communicative acts and their meanings independent of the content. This set aims for completeness, simplicity and conciseness. Although all agents are not required to implement all the pre-defined message types and protocols, FIPA ACL compliant agents must meet certain minimal requirements:

1. Agents must reply to messages they do not recognize or they cannot process with the phrase "not-understood". Sender agents must be able to handle the not-understood messages they receive.

2. When ACL compliant agents implement any subset of the predefined message types and protocols, they must abide by the semantic definition of the referenced act.

3. When ACL compliant agents use the name of communicative acts defined in the specification, they have to implement them according to their definition in the standard.

4. Agents may add new communicative acts not defined in the specification. They can do so provided the meaning of the new act does not match one of the already predefined acts. Nonetheless, it is the sender's responsibility to ensure that the receiving agent will understand the meaning of the act.

5. ACL compliant agents must be able to correctly, both encode messages for transport and decode messages from well-formed character sequences.

Contrary to KQML, FIPA ACL's semantics are formally defined using a Semantic Language (SL) [url97]. The semantics of a communicative act is defined through its rational effect (RE) and its feasibility preconditions (FP). The rational effect defines both the results of performing this communicative act and the conditions which must hold true at the recipient's end. The feasibility conditions characterize the conditions to be satisfied before an agent can plan to execute the communicative act. Feasibility conditions can be further subdivided into "ability conditions" and "context-relevance preconditions". Ability conditions characterize the actual ability of an agent to perform the communicative act whereas the context-relevance preconditions characterize the relevance of the act to the context it is performed in [url97]. Based on the rational effect, an agent may select a communicative act to perform but it cannot assume that the rational effect will necessarily follow.

Facilitation and registration primitives are treated differently in the two ACLs. While in KQML, they are first-class fully defined performatives, in FIPA, they are requests for actions with no formally defined specifications or semantics.

In their article "Agent Communication Languages: The Current Landscape", Labrou et al. [YLP99] analyze the actual situation with systems using KQML and FIPA ACL. From their perspective, systems that use any of the two ACLs must provide:

1. a set of API supporting the composition and the exchanging of ACL messages
2. services to support naming, registration and facilitation services
3. code for the primitives according to the semantics for that particular application

The first two items are typically reusable components whereas the third item has to be provided by the implementers. In reality, no such services have been promoted and put in place. Further, because existing semantic approaches can not easily be translated into code, the implementation of the reserved primitives according to the specifications is based on the implementers' intuitive understanding of the semantics rather than their concise definition. Labrou et al. [YLP99] conclude that the choice of an ACL is based on the extent of modalities such as belief, intention, desire implemented in the agent according the semantics of the agent's theory. ACLs with pragmatic concerns such as KQML are recommended for agents with low modalities implementations.

2.5 Ontology

The ontology problem has been considered secondary for a while and has not been tackled in all its magnitude. In its 1997 release, FIPA ACL did not address the ontology sharing problem although it is at the core of agent inter operability issue. Nwana and Ndumu [NN99] ask the question "is it realistic to expect knowledge and co-operation level interoperability without a significant degree of ontological sophistication of the agents concerned?" Fipa '98 defines the ontology sharing problem as "The problem of ensuring that two agents that wish to converse do, in fact, share a common ontology for the domain of discourse. Minimally, agents should be able to

discover whether or not they share a mutual understanding of the domain constants." [url98]

Systems like Cyc [Len95] have been built to facilitate inter-agent communication. They are based on general purpose ontologies. Since then, the trend has moved towards the development of domain-specific ontologies and ontology translators. The main reason for such a shift is that general purpose ontologies are bound to miss some intricacies of some domains. Further, for most applications, these ontologies are likely to be unnecessarily complex.

Some domains are common for a wide variety of tasks. To reduce the high cost of building ontologies, it is critical to encode the ontologies in a reusable form. Hence, the ontology of the application may be assembled from already defined ontologies available in ontology repositories. Ontolingua, KSE's solution to the ontology problem, has been developed from this perspective. It is a language to express ontologies developed at Stanford University. Its set of tools and services is not restricted to the development of ontologies by individuals, it also supports "the process of achieving consensus on common ontologies by distributed groups." Ontolingua's tools allow users to manipulate ontologies stored on an ontology server over the World Wide Web.

This approach of using ontology servers is endorsed by FIPA 98's Ontology Specification. Ontology servers are not necessarily confined to FIPA's domain or FIPA-compliant. Examples of available ontology servers are Ontolingua XML/RDF ontology servers [FFR93], ODL databases ontologies servers. To offer a standard view of the services provided by those different ontology servers, FIPA has introduced a new category of agents called ontology agents (OA). OAs advertise ontology servers and potentially provide translation services to FIPA agents.

With the development of domain-specific ontologies and their availability as building blocks for more application specific ontologies, the cost of constructing ontologies is amortized on different users and many users are able to benefit from reusing these modules. In this chapter, we have discussed the major issues of Multi Agent Systems and in the next chapter we introduce the notion of Negotiation Service Provider or NSP in the context of multi-agent systems.

Chapter 3

Design of NSP

In this chapter, we will present our proposed solution to the problem of negotiation in a heterogeneous multi-lingual multi-agent environment. Specifically, we will discuss the design issues of our suggested solution. In the first section, we will describe the model of our Negotiation Services Provider (NSP). In the next section, we will examine the modules and their interactions within NSP. We will conclude this chapter with a description of the different stages in providing the negotiation services under NSP.

3.1 NSP Model

Providing negotiation services within the stipulated requirements is based on three major foundations: communication, negotiation protocols and agent communication languages. In order to offer its services, NSP must be able to establish a link and communicate with other entities such as agents and other servers. An agent communication language becomes essential.

The model we propose for NSP is depicted in Figure 7. It consists of the following four major modules: the Negotiation Protocols Suite (PS), the Agent Communication Language Suite (ACLS), the Negotiation Agent Generator (NAG), the Negotiation Agent (NA) and the Management module (MM).

- Negotiation Protocols Suite (PS):

  The Protocol Suite consists of a set of negotiation protocols and a collection of methods. These methods can be grouped around three responsibilities:

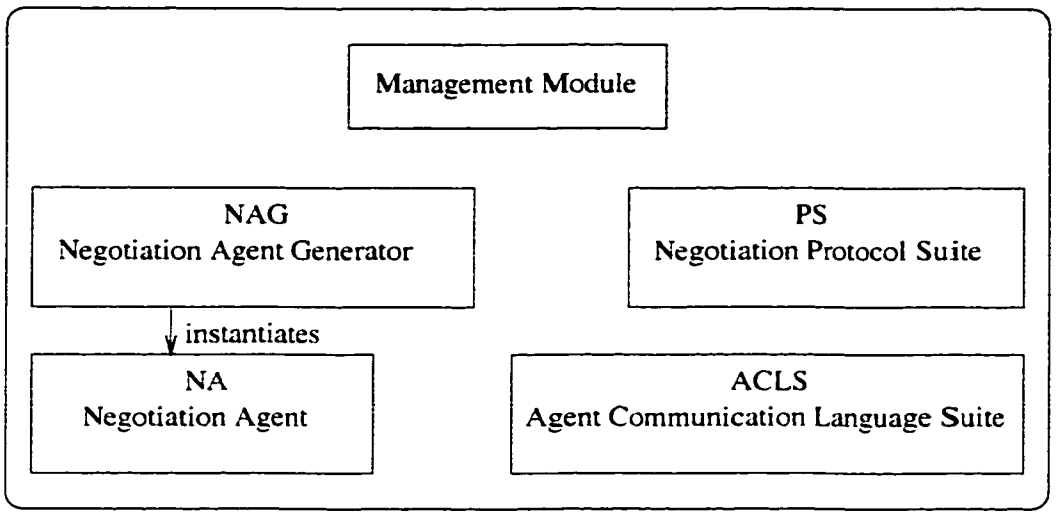

Figure 7: Negotiation Model

1. Manage protocols by introducing new protocols to the suite and maintaining their evaluation.
2. Manipulate and manage the implementations of the protocols.
3. Interact with the other modules of NSP by providing the ratings and implementations of protocols and by initiating the maintenance and update patches it receives.

Each protocol has its implementation in terms of lines of code and is identified using a set of characteristics. This information is represented in a two dimensional table as in table 1 below.

Protocol Name	Function Name	Dynamic Environment	Optimal
Contract Net	Cnet()	10	0
Multistage Negotiation	Multistage()	10	4
Partial Global Planning	PGP()	3	4

Table 1: Sample Classification of Protocols in the Negotiation Protocols Suite

The "Protocol Name" field represents a unique identifier for the protocol. The "Function Name" field holds the pointer to the function that implements the corresponding protocol. There are only two parameters defined in table 1, namely dynamic environment and optimal. So each protocol defined in the

suite is rated based on these two criteria on a scale from 0 to 10. A rating of 0 is the minimum and means that the protocol does not provide any support for this criterion. A rating of 10 is the maximum and thus this protocol is the best qualified for a situation that requires this criterion. So in this table, the Contract Net protocol supports a dynamic environment but is not optimal.

In general, the number of characteristics would be larger than two. In fact, the more characteristics are specified, the better a protocol is defined, thus the better match between a protocol and a conflict to be handled could occur. At the same time, an extended set of characteristics potentially hinders the protocol selection process. One guideline to build a balanced set is to select the characteristics the most likely to be required in the context of conflicts.

Further, this table is not static and is expected to grow in both rows and columns as new protocols are integrated. Adding a new protocol grows it vertically. Adding a new criterion grows it horizontally. Depending on the information extracted, the table is traversed differently. A search for protocols with specific characteristics requires a vertical and then a horizontal traversal to identify the appropriate one. If the other agent involved in the conflict imposes a specific protocol, the table is traversed horizontally to provide the pointer to its implementation.

- Agent Communication Languages Suite (ACLS):

  Due to the emergence of different agent communication languages (ACL) in the field of agenthood, NSP must have provision to communicate using different ACLs. ACLS holds both ACL translation engines and methods. Each new ACL available in NSP must have a translation engine in the ACLS. The translation engine of ACL1 takes a primitive C1ACL1 in ACL1 and translates it to the corresponding primitive C1 in the NSP's native ACL. This translation is done at the level of the semantics and the syntax of the two ACLs. The collection of translation engines can be conceptualized as a table. The content of a cell holds the syntax of the command. Semantically equivalent commands are listed on the same row. Columns identify the language of the command. For example in the table 2 below, the internal command C1 is equivalent to the command C1ACL3 in the language ACL3, whereas the internal command C3 is equivalent

to the command C3ACL1 in the language ACL1.

Internal Command	ACL1	ACL2	ACL3
C1	C1ACL1	C1ACL2	C1ACL3
C2	C2ACL1	C2ACL2	C2ACL3
C3	C3ACL1	C3ACL2	C3ACL3

Table 2: Commands in Different ACLs.

As table 2 shows, there are two approaches in using this table. To map a command C1ACL1 in a specific language ACL1, the table is traversed horizontally to reach this language's column, ACL1, then horizontally to map it to its equivalent internal command C1. Alternately, to translate an internal command C3 into another language ACL2, the table is traversed vertically to find the internal command's row, C3, then horizontally to map into its equivalent command C3ACL2 in ACL2.

Table 2 changes as new ACLs are introduced by providing the semantics and the syntax of the new commands. Based on these two parameters and the language it belongs to, the command is inserted in the table. If no internal command is equivalent to the new command, an extension of NSP's native language has to be extended to accommodate the external extension.

ACLS is necessary if NSP has to handle different ACLs. However, it is quite possible that one server in an agent society becomes an ACL-translation services provider. In this case, we recommend "outsourcing" ACLS. This approach is depicted in Figure 8. In this setup, NSP provides ACLSP with the command NSPC1 and specifies the language ACL1 to be used. It receives from ACLSP the command ACLC1 to issue in the specified ACL, ACL1 as shown in Figure 8. Depending on what this server is able to translate, NSP might not be able to submit the internal command NSPC1 for translation as is, instead, it will have to translate it to a universal command C1 supported by the services provider before getting its equivalent ACL1C1 in the ACL it originally specified, namely ACL1 as illustrated in Figure 8.

- Negotiation Agent Generator (NAG):

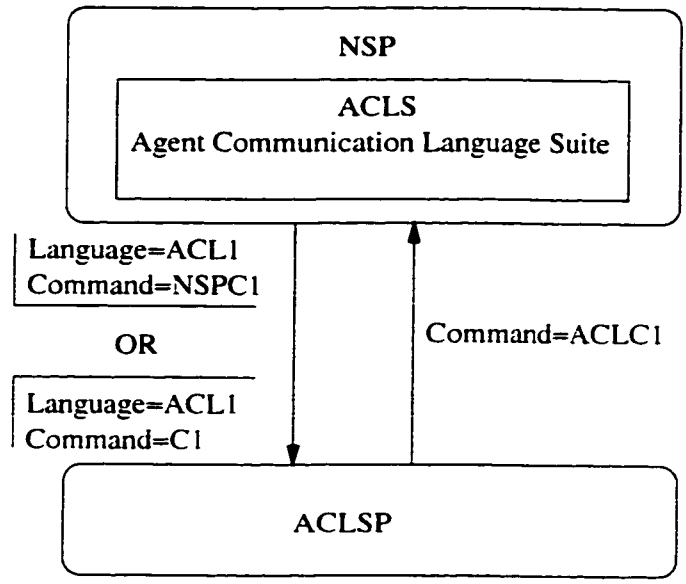

Figure 8: Communication between ACLS in NSP and ACLSP

NSP spawns one Negotiation Agent NA for each request received from a client agent and delegates to it the task of conducting the negotiation. NAG is the module that handles this activity. This module is also responsible for the management of the spawned agents. The management tasks include the creation and destruction of the spawned agents as well as their supervision. Being the agent generator, NAG maintains all the components necessary to generate an agent.

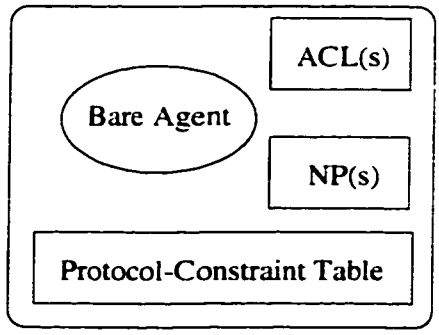

Figure 9: Negotiation Agent Generator

NAG holds the skeleton of a bare agent as illustrated in Figure 9. When NAG generates a new NA, it uses the bare agent to create a generic "intelligent software agent". Using its ACL(s) and NP(s) modules, it customizes the generic

agent to be a negotiation agent with its own specific agent communication language and negotiation protocol. Both ACL(s) and NP(s) are respectively subsets of ACLS and PS.

Depending on the strategy adopted in supplying the NA with agent communication languages and negotiation protocols, ACL(s) and NP(s) in Figure 9 hold different data. There are two possible alternatives to supply NA with agent communication languages and negotiation protocols. Depending on the strategy, ACL(s) and NP(s) in Figure 9 hold different data.

In one strategy, NAG conducts a preliminary dialog with its client. Hence, it is able to supply NA with the adequate ACL and the appropriate negotiation protocol. In this case, ACL(s) and NP(s) are responsible for contacting ACLS and PS respectively to extract the needed information from them.

In the second strategy, NA is assigned the case of a client without any preliminary investigation on the conflict. NA is provided with pre-selected language(s) and protocol(s). Thus, NAG maintains local copies of them and both ACL(s) and NP(s) hold this information respectively.

NAG also maintains a compact copy of the Protocol-Constraint Table (PCT). Table 3 is a sample of this table illustrating an example protocol selection. Since the implementations are internal to PS, when the table is exported, as it is the case in this version of PCT, the pointers to the implementations are removed.

Protocol Name	Dynamic Environment	Optimal	Competing Agents
Contract Net	10	0	0
Multistage	10	4	0
Partial Global Planning	3	4	5

Table 3: Protocol-Constraint Table.

Given one constraint, this table provides the best suited protocol available through NSP to conduct the negotiation. Assume, NA needs to find the most appropriate protocol for a conflict with *competing agents*. NA traverses the table horizontally looking for the column of *competing agents*. Then NA traverses the column looking for the highest rating. In this case, 5 is the highest

rating and it is assigned to *Partial Global Planning*. Therefore, *Partial Global Planning* is the most appropriate protocol for *competing agents*.

However, conflicts are typically not restricted to one constraint. Instead they carry with them a group of constraints. For each protocol, its rating based on each of these constraints is added to compile a mean rating against this group of constraints. Then the simple traversal applied in the case of a single constraint is applied on the mean rating to produce the final results. Consider the scenario where the conflict to negotiate has two constraints *optimal* and *competing agents*. To illustrate this algorithm, assume that the protocol-constraint table is augmented with one column titled "Rating List". Each cell in this column is initiated with a rating of zero. Table 4 illustrates the table in the Phase 1 of the algorithm.

Protocol Name	Optimal	Competing Agents	Rating List
Contract Net	0	0	0
Multistage	4	0	0
Partial Global Planning	4	5	0

Table 4: Multiple Constraints' Conflict - Phase 1

For each constraint, the table is traversed. The rating of each protocol based on this constraint is added to its "Rating List" corresponding cell. For the column of *optimal* is traversed. The rating of *Contract net* is added to its "Rating List". Hence, the "Rating List" for *Contract net* is equal to 0 (0+0). The "Rating List" for *Multistage* is equal to 4 (0+4). For *Partial Global Planning*, it is equal to 4 (0+4). Table 5 shows the table at this stage.

Protocol Name	Optimal	Competing Agents	Rating List
Contract Net	0	0	0
Multistage	4	0	4
Partial Global Planning	4	5	4

Table 5: Multiple Constraints' Conflict - Phase 2a

The column of *competing agents* is traversed next. Hence, the "Rating List" for *Contract net* becomes equal to 0 (0+0). For *Multistage*, it is 4 (4+0). As for

Partial Global Planning, it is 9 (4+5). Table 6 shows the table at the end of Phase 2.

Protocol Name	Optimal	Competing Agents	Rating List
Contract Net	0	0	0
Multistage	4	0	4
Partial Global Planning	4	5	9

Table 6: Multiple Constraints' Conflict - Phase 2b

Once all the constraints have been processed, the protocol with the highest accumulated rating in the "Rating List" is the best suited protocol from the group of constraints' perspective. From table 7, the highest rating is 9 and it corresponds to the *Partial Global Planning* protocol.

Protocol Name	Optimal	Competing Agents	Rating List
Contract net	0	0	0
Multistage	4	0	4
Partial Global Planning	4	5	9

Table 7: Multiple Constraints' Conflict - Phase 3

Note that if the conflict requires an *optimal solution*, then either *Multistage* or *Partial Global Planning* is appropriate.

- Negotiation Agent (NA):

  Each request for negotiation issued by a client agent is handled by one NA. This NA is created for this purpose only and is destroyed once its mission is accomplished.

  NA is a generic agent augmented with a couple of tools as illustrated in Figure 10. An agent created accordingly can communicate on the network both at a low level using a network protocol such as TCP/IP and at a high level using an agent communication language such as KQML. As a Negotiation Agent, it includes some negotiation protocols. Further, NA holds a copy of the Protocol-Constraint Table (PCT) maintained by NAG. NA preserves contact with the NSP that created it and reports back to it.

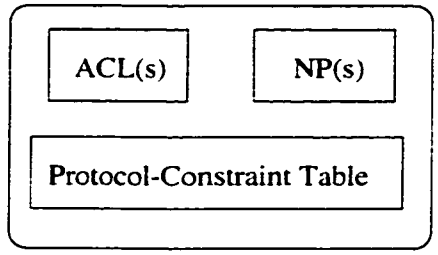

Figure 10: Negotiation Agent

The distribution of the ACLs and the negotiation protocols between the NAs and the NSP may change from one implementation to another. In one approach, NSP may be centrally organized with very tightly coupled NAs. Alternately, NSP may have a distributed organization with loosely attached NAs. Depending on the size of the community, one distribution is more advantageous than another. In a society with hundreds of conflicting agents, the centralized approach is a viable solution, however, as soon as the targeted society grows to higher orders, the distributed strategy is necessary to ensure the survival of any NSP under such loads.

A negotiation agent could be multi-lingual and have the ability to use all the negotiation protocols known to NSP. We propose three approaches with respect to ACLs in NA.

IC	ACLi
C1	C1ACLi
C2	C2ACLi
⋮	⋮
Cn	CnACLi

Simple ACL

IC	ACL1	ACL2	ACL3
C1	C1ACL1	C1ACL2	C1ACL3
C2	C2ACL1	C2ACL2	C2ACL3
⋮	⋮	⋮	⋮
Cn	CnACL1	CnACL2	CnACL3

Hybrid ACL

IC	ACL1	····	ACLn
C1	C1ACL1	····	C1ACLn
C2	C2ACL1	····	C2ACLn
⋮	⋮		⋮
Cn	CnACL1	····	CnACLn

Complete ACL

Figure 11: Different ACL Strategies for Negotiation Agents

1. **Simple Approach**: This approach provides the instantiated NA with the one and only ACL specified by the client agent in the initial contact between this agent and NSP. This approach is illustrated in the left most

table of Figure 11. In this case, the Agent Communication Languages Suite's table 2 is basically reduced to two columns: the Internal Command column and the column of the ACL which the client agent uses as specified in the initial contact between this agent and NSP.

This model is particularly suited when the Agent Communication Languages Suite is implemented as a collection of translators. With this approach, the translator of the required language is simply passed along to the NA. Furthermore, the simplicity of this approach puts less burden on the NA, reduces its size and the resources it needs. Thus, spawning NAs is done much faster.

2. **Complete Approach**: This approach provides the generic NA with the complete Agent Communication Languages Suite's table 2 as depicted in the right most table of Figure 11. NA is fully multi-lingual from NSP's perspective.

   This approach is advantageous because:

   - An important number of conflicts occur because of the lack of proper communication due to the inexistence of a common communication language. Chances are NA will have to use different ACLs to resolve conflicts.
   - NAs and NSP maintain a loosely coupled relationship. More resources are freed up allowing NSP to maintain its quality of services.

3. **Hybrid or Balanced Approach**: In this approach, the most popular ACLs are initially supplied to the NA. This alternative is illustrated in the middle table of Figure 11. NAs are loosely coupled with their NSP and they can handle a large number of conflicts without having to download a less common ACL. Thus, this strategy retains all the advantages of the complete approach described above. Furthermore, by eliminating the ACLs the least likely to be used, the resources and the time taken to spawn a NA are also reduced giving this approach an edge over the complete one.

The above three approaches are very similar to the negotiation protocols organization mentioned below.

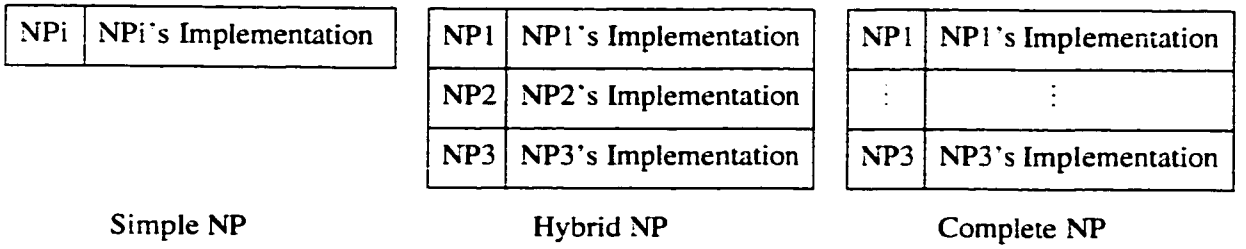

Figure 12: Different NP Strategies for Negotiation Agents

1. **Simple Approach**: In this approach, the NSP selects from its collection of protocols the most appropriate protocol to negotiate the conflict and provides the NA with its implementation. Therefore, it becomes the responsibility of NSP to get the initial problem description from the client agent and then to select the suitable protocol for negotiating this conflict. Relying on NSP to get the problem specification is a major setback for this approach since this task can be quite time and resource consuming. Such a distribution of tasks may be acceptable in a restrained environment. In an unbounded society, this approach is not an option.

2. **Complete Approach**: In this approach, NA is augmented with the complete negotiation protocol suite including the implementations of each supported protocol. In this setup, NA is largely independent of NSP and has all the necessary tools it needs to effectively conduct the negotiation. This strategy is quite powerful in a dynamic and complex setup where NA has more potential to switch protocols within the same conflict without relying on NSP's interference.

3. **Hybrid or Balanced Approach**: With this scheme, only the most general protocols are initially included in the NA. This approach is lighter than the complete one with respect to complexity and resources. The selection of its protocols produces a quite effective NA able to cover the majority of the conflicts with no external help in convoluted negotiation.

- Management Module (MM):

  This module handles the initial contact between the client agent and NSP. It saves the preferences of each client. It also maintains the history of the past negotiations the NSP performed, specifically the rationale behind the decisions

taken by the NA in the process of negotiating. To efficiently manage this knowledge, a database is maintained. The database holds the following information:

1. **Session ID**: Every time a client successfully contacts NSP and a session is established, a unique identifier is attributed to this connection for logging purposes.
2. **Customer identification**: It is represented by a combination of its IP address and its process ID.
3. **Decisions**: In the process of negotiation, NA is bound to adopt decisions.
4. **Rationale**: In the decision process, NA follows a rationale.
5. **Failure Causes**: NSP may fail to serve its client because it doesn't use a specific ACL or because the conflicting agent is capable to negotiate using a new protocol.

In the occurrence of converting NSP into a commercial product, further information needs to be maintained related to the billing process. The failure causes are used when NSP is updated.

These modules are the building blocks of the NSP. Each module handles a different aspect of NSP. They interact and cooperate to provide negotiation services in a dynamic heterogeneous environment through a standard interface.

3.2 Module Interactions within NSP

Upon the contact of an agent, NSP's different modules will interact in the process of providing the advertised negotiation services. These interactions are illustrated in Figure 13. NSP's modules can be grouped according to two categories: interactive with the client agent and non interactive. Dotted arrows show the interactions of the modules external to NSP whereas solid arrows show the internal interactions. The Management Module, the Negotiation Agent Generator and the Negotiation Agent interact with the clients directly. They represent the frontline of NSP with regard to the clients. Both the Negotiation Protocol Suite and the Agent Communication Language Suite belong to the non interactive category and they are shielded from the

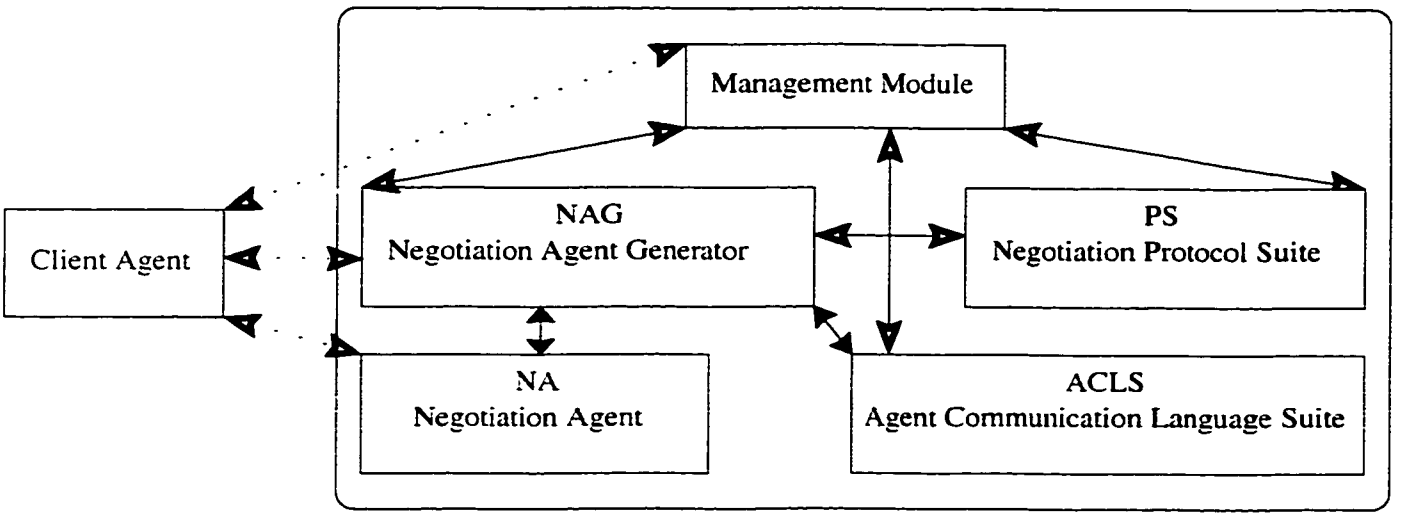

Figure 13: Interactions among the different parts

clients. In Figure 13, they are shown shaded. They provide the basic tools which allow NSP to provide these specific services of multi-lingual negotiation to its community.

The Management Module is the coordinator among the majority of modules. It communicates with the clients, ACLS, PS and NAG as depicted in Figure 13. It receives the connections of clients, and then directs them to the NAG module which handles the details of the negotiation. It is also responsible for initiating the online maintenance of both suites. Adding new protocols or ACL as well as updating them is routed through the Management Module. This module maintains the history and profiles of past clients as well as records of previous sessions. Therefore, it collects from NAG all the reports generated by NAs.

The Negotiation Agent Generator is at the center stage. It is the only module which communicate with all the others. It receives sessions established by the Management Module and hands them back when the negotiation is concluded along with the corresponding information produced. By spawning negotiation agents, NAG naturally has a special relationship with them and maintains a permanent connection with them. NAs rely on it to provide them with the ACLs and the negotiation protocols they need in the process of conducting their tasks. Therefore, NAG interacts with both ACLS and PS to retrieve new ACLs and protocols required by NAs.

The Negotiation Agent is under NAG's control. It communicates with only the client agent and NAG. In fact, NAG is the only link between the negotiation agents and NSP. All NA's requests for new ACLs and negotiation protocols are routed

through NAG. NA transmits to NAG any errors encountered or updates to the client's profile surfacing within the process of negotiating. NA relies on NAG to supply the ACLs and negotiation protocols necessary to pursue its task.

The Negotiation Protocol Suite is isolated from the outside world by both the Management Module and NAG. It receives negotiation protocols additions, upgrades, modifications from the Management module and applies them. From NAG, it receives requests for the transmission of various protocols' implementations. PS supplies these implementations to NAG which in turns transmits them to its NA.

The Agent Communication Language Suite has similar interactions as PS with the other modules of NSP. It communicates only with the Management Module and NAG. It receives upgrade and modification instructions from the management module. It also fulfils requests for different ACLs required by NAs through NAG.

Therefore a session is built on top of effective interactions among the different NSP's modules. Each module is assigned specific tasks and responsibilities. Communicating together, they are able to efficiently provide different flavors of negotiation services in a multi-lingual dynamic environment. The managerial activities are basically shared between both the Management Module and the Negotiation Agent Generator. In a compact implementation of NSP, these two modules may be merged together.

3.3 Agent's Initial Contact with NSP

The first encounter of an agent with a NSP establishes the basis for interactions with its community. To successfully exchange these initial messages, a common language and a link are required. Without a commonly understood language, no contact can be established. Without knowing where to direct the information, no link can be built. Hence, for a message to reach its destination and be properly interpreted, it should have the following:

- Know the destination: Depending on the way the agent discovered NSP, it would know the destination differently. The agent might have the direct address because NSP advertised its services earlier. Therefore, the message is sent straight to NSP's address. Another possibility is that the agent discovered NSP through the services of a facilitator and hence, it only knows the qualifier used

by this facilitator to address this NSP. In this case, the message is directed to this facilitator and would contain the qualifier. Then, it is the facilitator's responsibility to uniquely map this qualifier to an address and then properly deliver the message to this address.

- Have a standard format: The most widely used standard format for messages is the one used by network protocols such as TCP/IP. Once a low level communication is established, a higher level communication language could be agreed upon if both parties understand and speak this language. At the foundation of the Internet, TCP/IP protocols suite is the standard communication tools used to establish the basic first contact between NSP and its clients.

  From the TCP/IP protocol suite, two protocols are possible candidates with potential to support the requirements and the procedure for establishing this initial contact. Both of these protocols work over particular ports. One possibility is to use the User Datagram Protocol (UDP). UDP is a connectionless simple protocol with very little overhead. The other candidate is the Transmission Control Protocol (TCP). TCP provides the reliability and efficiency lacking in UDP. It also establishes a virtual circuit between the server and the client.

There are two possible scenarios to secure the initial link. In one scenario, one specific TCP and/or one UDP port is dedicated to NSP's services. An agent would talk to this specific port and transmit the reference to the agent communication language of its choice to the server in the first packet its sends. The server receives this message, recognizes the language, sends an acknowledgement to the agent and pursues the conversation in this specific language. This scenario can handle standard languages.

The alternative scenario is to dedicate a range of ports for this service. Depending on which port the agent contacts, an agent communication language is implicitly selected. For example, the interval 1300-1800 can be reserved. A different language is coupled to each of the ports. KQML is coupled with 1300, FIPA ACL is attached to 1301, etc. If the agent connects to port 1300, NSP and this agent would communicate using KQML. Although the number of languages is restricted to a maximum of 500 in our example, this strategy reduces the overload on one specific port and distributes it over a large spectrum.

A two-way communication protocol is followed to connect agents to a NSP. In the first step, NSP advertises its services and its contact address to the community. Hence, the destination of NSP becomes available to the community. All agents receive this broadcast and depending on their function and capabilities, they potentially save this information in their knowledge base for future use. Two categories of agents are identified in the agent community:

- Long Term Agents: These agents have a long life span. Typically, they live long enough to receive NSP's broadcast messages and save them in their own knowledge base. They are able to directly access NSP services. Facilitators, yellow and white page maintainers belong to this category. A Directory Facilitator (DF) within FIPA is defined as "an agent which provides a "yellow pages" directory services for the agents. It stores descriptions of the agents and the services they offer." [url97] Similarly in KQML, a Communication Facilitator is "an agent that performs various useful communication services, e.g. maintaining a registry of service names, forwarding messages to named services, routing messages based on content, providing "matchmaking" between information providers and clients, and providing mediation and translation services." [FLM97]

- Short Term Agents: Agents with short life span belong to this category. Hence, they typically miss NSP's broadcasts. They could also be lacking the resources to keep track of the offered services. These agents heavily depend on the "long term agents" to communicate to them the addresses of NSPs. Specialized agents are examples from this category.

With its advertisement, NSP essentially targets "long term agents". "Short term agents" will consult the services of "long term agents" to identify a NSP to contract. In highly dynamic and rapidly expanding environments, NSP may broadcast its advertisement regularly to keep new long term agents uptodate.

In the second step, the agent responds to NSP's advertisement. The message will proactively initiate the ACL selection process between the agent and NSP. The agreement on an ACL concludes the preliminary communication and both NSP and its client can proceed to solve the client's conflict.

3.4 Event Trace

We make use of a sample scenario to illustrate the event trace in the interaction between NSP and its client. The scenario initially starts in a heterogeneous dynamic environment with an agent facing a conflict. Consider Figure 14 as an illustration of such a environment. In this Figure, NSP and the agents are depicted as rectangles with rounded corners. The Agent Communication Language Suite, the Negotiation Protocol Suite, the Management Module and the Negotiation Agent Generator are clearly marked within NSP. Each agent is coupled with the ACL it understands. Agent A3 uses the agent communication language ACL3 whereas Aj uses ACL1. All these agents are developed by different people with dissimilar purposes. Agents are dynamically created and destroyed in this setup.

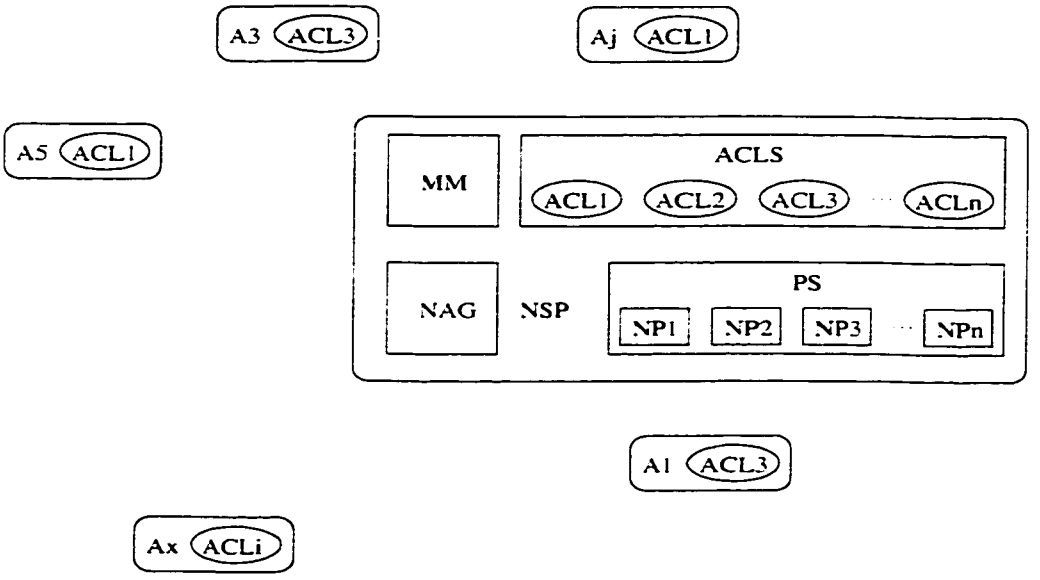

Figure 14: Typical Heterogeneous Dynamic Environment.

The pre-condition for any negotiation is that a conflict should exist. Hence, a conflict is the trigger for any interaction with NSP. Detecting a conflict is a crucial milestone in the process of achieving a task or a goal. An agent typically detects its conflicts when some requests it issued are rejected or it can not accommodate requests it received because satisfying them interferes with its own goals. Recognizing conflicts is beyond the scope of this research.

To solve a conflict, the agent decides to negotiate with the conflicting agent. For this purpose, a negotiating agent is used. In an agent society, NSP plays the role of

providing a negotiating agent with suitable characteristics. Agents contact NSP as shown in Figure 15.

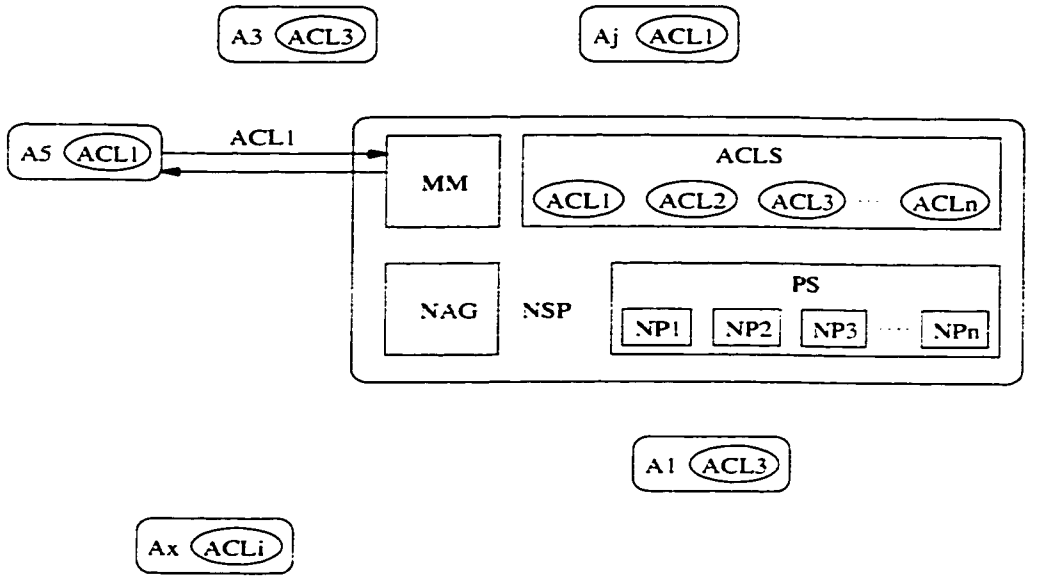

Figure 15: Agents establishing links with NSP

The events corresponding to interactions illustrated in Figure 15 are traced in Figure 16. Each event is assigned a number. For example, e1 is event #1. NSP advertises its services with all the facilitators and yellow/white pages directories services [e1]. The agent requiring specialized negotiation services, agent A5, searches for a NSP within its community [e2]. Once it finds the address of the NSP [e3], agent A5 attempts to establish a link with it as described in the previous section. A5 and NSP establish a basic connection at the network level [e4], and agree on the agent communication language (ACL) to use for the duration of this session, ACL1 [e5].

Since NSP is "multi-lingual", the agent selects its preferred ACL. Typically, this ACL will be supported in NSP. In Figure 15, A5 speaks the agent communication language ACL1 which is supported in NSP. Therefore, ACL1 is selected. If this ACL is not so common and hence is not recognized within the NSP, another ACL has to be agreed upon by both parties. Further, NSP keeps a record of this failure as a future desired enhancement in its maintenance log.

Upon the success of the first encounter, the communication with A5 is handed to NSP's Management Module (MM) [e6]. MM starts an official session between NSP and the client agent, A5. It attributes to it a unique identification number. It also

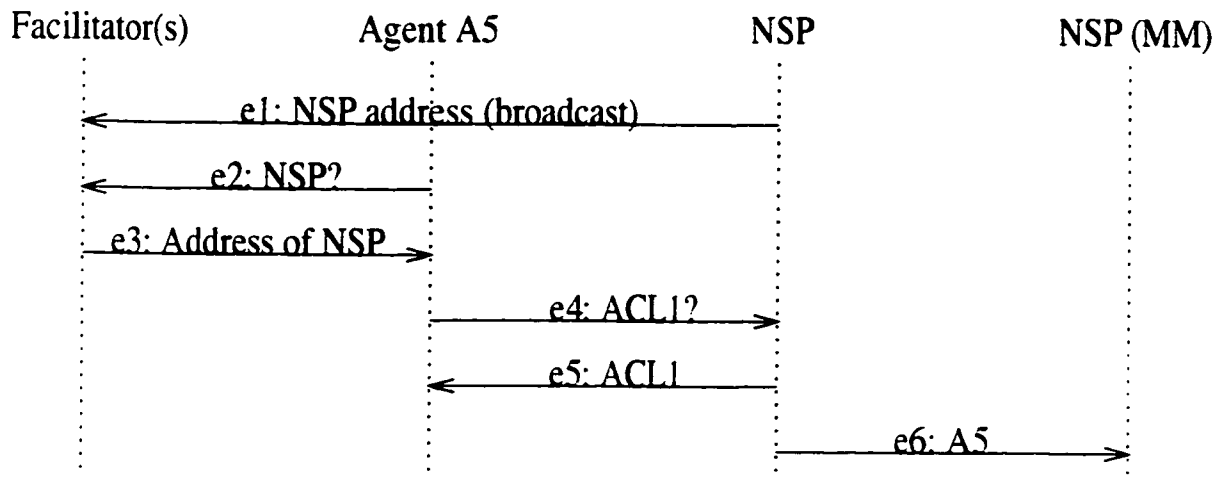

Figure 16: Event Trace of the First Contact.

creates a new record in its knowledge base to log the details of this session. Then it retrieves the preferences and the history of agent A5 it has gathered through their previous encounters. MM hands the communication with A5 to NAG along with all the relevant information available within NSP. NAG takes over this session as shown in Figure 17.

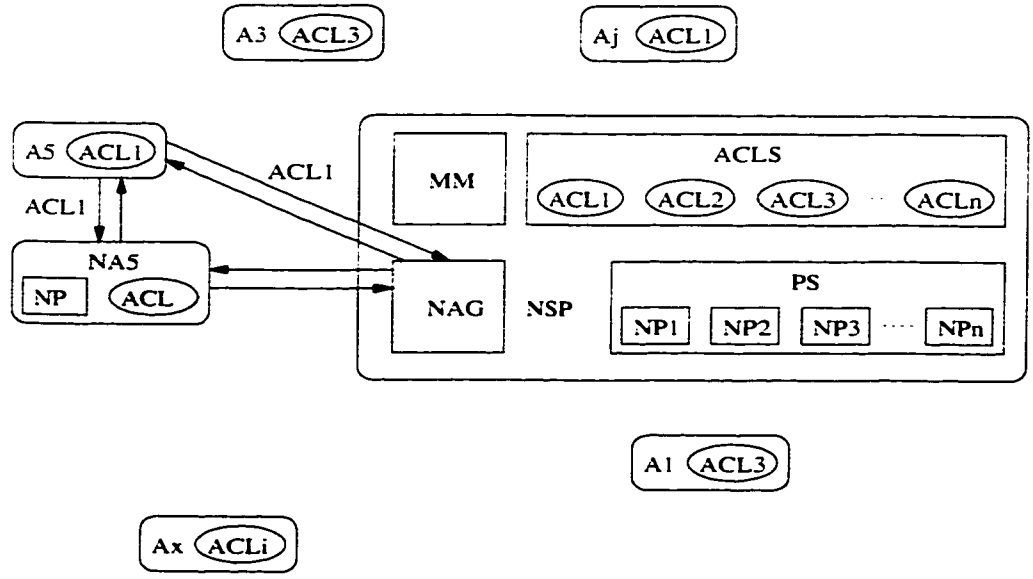

Figure 17: NSP's Negotiation Agent Generator spawns Negotiation Agents

This communication transfer from MM to NAG corresponds to event [e7] traced in Figure 18. NAG spawns a negotiation agent NA5 and transfers the connection it

has with the client, A5, to the negotiating agent, NA5 [e8]. This 3-way connection is illustrated in Figure 17. In NA5, NP represents the negotiation protocol approach whereas ACL the agent communication language strategy adopted in instantiating NAs. Although NA effectively controls the session, it reports back to NAG and thus maintains a contact with it.

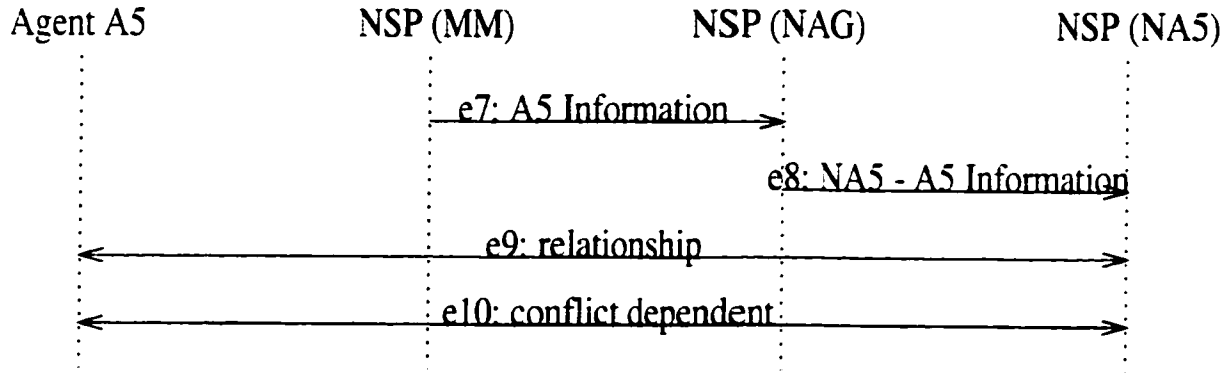

Figure 18: Event Trace of the Preliminary Setup for Negotiation.

Once NA5 is attached to the client agent, A5, it starts by establishing the general guidelines of their relationship [e9]. Event e9 has a bi-directional arrow representing a potential loop in the exchange between A5 and NA5. The level of authority delegated to the negotiating agent is a crucial issue addressed at this stage. These levels are described in table 8. When the level is minimal, the client directly supervises NA and makes all the decisions. NA effectively becomes an interface between the client and the conflicting agent. In the complete level, the client delegates all the decision making powers to NA. With the custom authority, the client and NA identify some areas or subjects where NA has full control and others which are reported to the client for a decision. For example, a client would decide to keep the control of its budget. Hence, if changes are required to the budget, NA has to consult with its client before making any commitments.

Other issues related to the client's requirements, the limits of its potential commitments, its expectations, its goals are also defined from a global perspective in the exchange between NA5 and A5 [e9]. Using the past knowledge about general preferences of A5, NA5 confirms them again and adds any new modifications required. Depending on whether these modifications are permanent or just related to the current session alone, it updates its own copy of this client's profile and transmits this

Authority Level	Definition
Minimal	Client takes the decision
Custom	Client and NA distribute authority depending on subject
Complete	NA takes the decision

Table 8: Levels of Authority of NA

update to NAG [e9]. A sample of such profile is illustrated in table 9.

Identity	A5
Level of authority	Custom
Description	Cooperative
Domain	E-Commerce
Goal	Maximize Profit
Preference	Fast Negotiation

Table 9: A5 Sample Profile

Next, NA5 attempts to collect further information from A5 regarding the current situation to build the conflict dependent preferences of its client [e10]. It starts by building a description of the conflict in terms of its cause, context, its client's prioritized list of goals, constraints, margin of maneuver, information. It also seeks to get an evaluation of their relative importance from the client's perspective. At this stage, the negotiation agent is ready to contact the other agents involved in the conflict for two reasons: to complete its view of the cause of the conflict and to attempt to negotiate a settlement suitable to all parties.

NA initiates the communication with the conflicting agents. In the scenario depicted in Figure 19, NA5 contacts Ax. In the case where more than one agent is involved in the conflict, NA contacts them all. The event trace of this stage are in Figure 20. NA5 attempts to find out what are the ACLs that Ax understands [e11] and [e12]. NA5 might have to contact NAG to get the ACL they adopted [e13] and [e14] depending on the ACL strategy in building negotiation agents in NSP. If the ACL is not supported by NSP and there is no other alternative ACL suitable for all parties, then NA5 has no other option but to withdraw from the negotiation and inform both its client, A5 and NAG of the outcome. NAG generates an entry for this

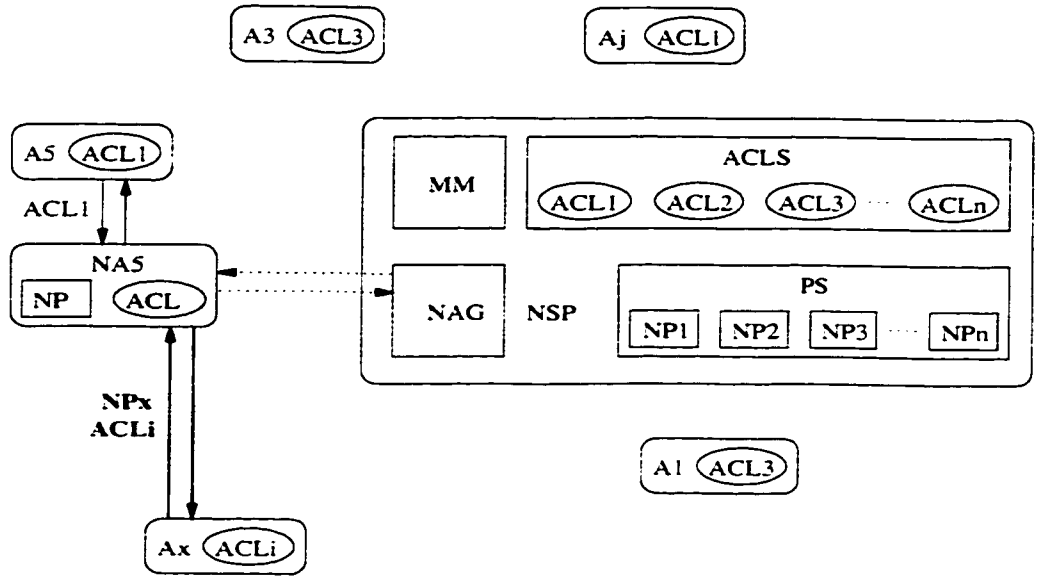

Figure 19: Negotiation Scenario in NSP

failure in the maintenance log. Once, they find a common ACL, they can start the negotiations. Since ACLi is available in NSP, the negotiation between NA5 and Ax is pursued using ACLi. NA5's first task is to investigate the cause of the conflict from Ax's perspective [e15]. A bi-directional arrow is used with event [e15] to show the possibility of an exchange of events between NA5 and Ax.

Given all the information it gathered, NA5 consults the Protocol-Constraint Table (PCT) and selects the most appropriate protocol to use for such a conflict. Suppose it is NPx. If NA5 does not have the implementation of NPx, it requests it from NAG [e16] and [e17]. NA5 launches the negotiation with Ax using NPx [e18a]. Within the application of NPx, NA5 communicates with A5 as often as it is set by the level of authority. NA5 reports to A5 the status of the negotiation and consults A5 to check if any changes have happened in A5's context that might affect the outcome [e18b].

Depending on the strategies followed in building a negotiation agent, the order of these events might be altered. With the simple approach, NA is equipped with only one negotiation protocol and one ACL. In this case, the preliminary investigation is conducted prior to the spawning of NA. Typically, NAG would take over these steps and spawn NA with the adequate protocol and ACL for the conflict it will handle.

Figure 19 illustrates the situation where the other agent involved in the conflict, in this case Ax, is able to conduct the negotiation without external help. NA5 directly

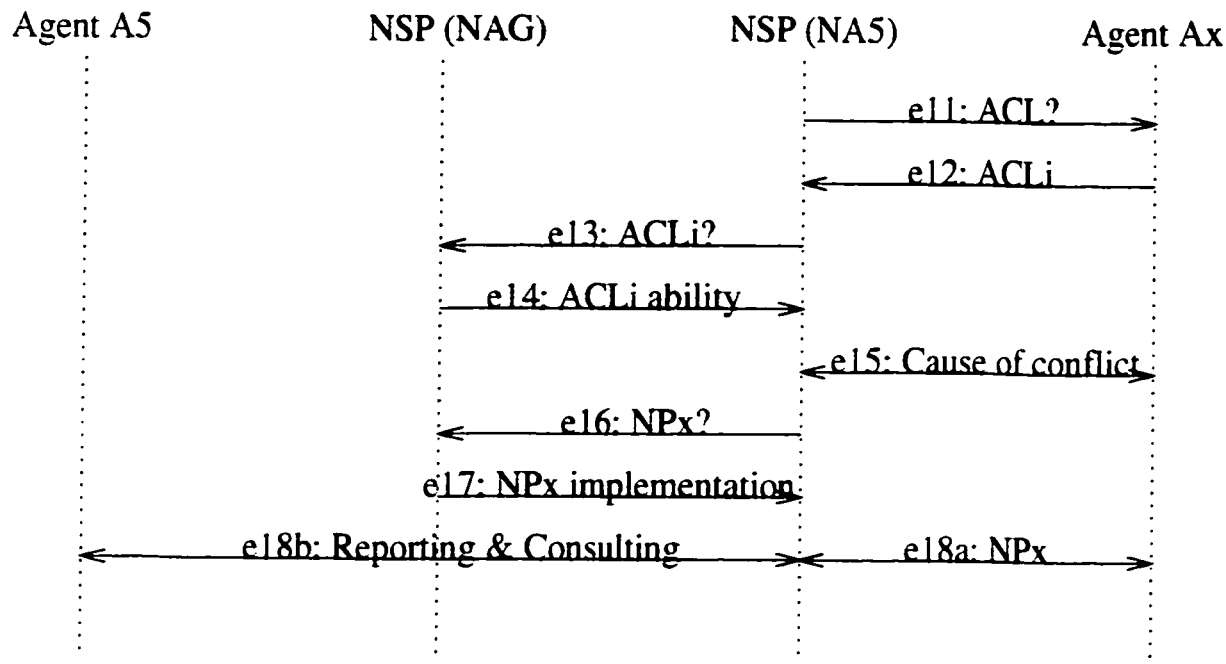

Figure 20: Event Trace of the Negotiation.

contacts it.

Conflicting agents need not to know how to negotiate since they, too, can resort to the services of NSP and have their own NA negotiating on their behalf. Figure 21, illustrates this scenario.

Ax and A5 are facing some conflicts. Both Ax and A5 resort to the services of NSP. Agent Ax contacts NSP and establishes a link using ACLi independent of A5. NAG spawns one NA for A5 and another one for Ax. NA5 and NAx are created for A5 and Ax respectively. NA5 and NAx conduct the negotiation for resolving the conflict between A5 and Ax on the behalf of their clients. Both NAs contact each other and attempt to select a common ACL. Since both of them are spawned by the same NSP, they use the same Internal Commands (IC). To communicate among each other they opt for IC and hence save the translation overhead. The event trace of this scenario is the same as the one in Figure 20 except Agent Ax is replaced by NSP (NAx).

While negotiating, NA has to maintain a record of certain critical stages. For each phase, it has to log:

- All the alternatives

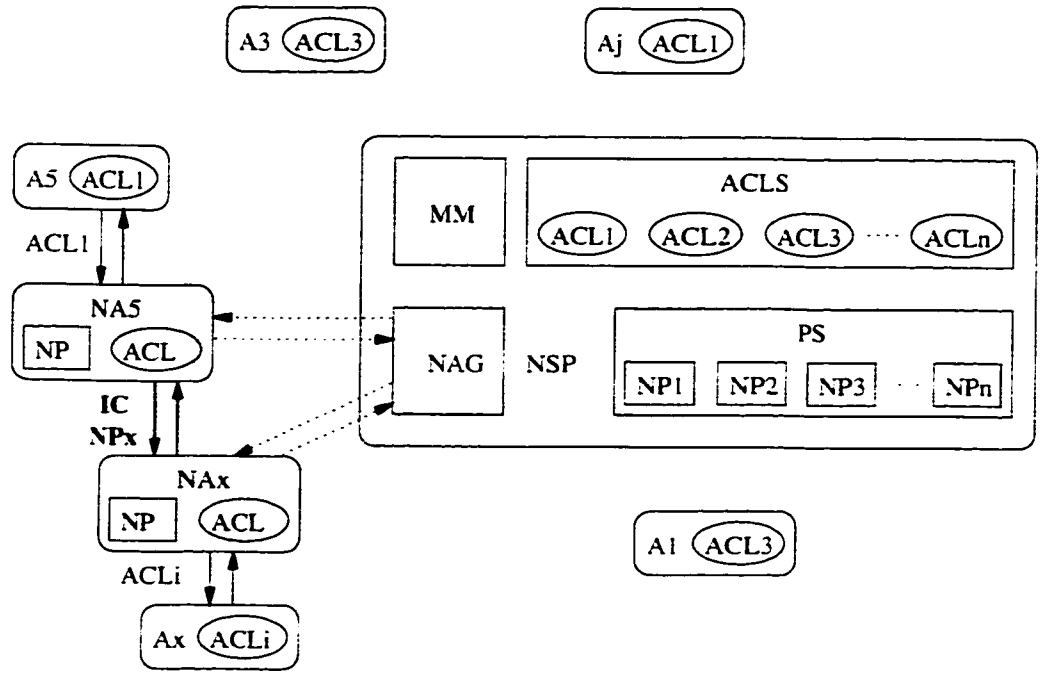

Figure 21: Alternative Negotiation Scenario in NSP

- The decision selected
- The rationale followed in the decision process

According to the level of authority (see table 8) specified by the client agent in its preferences, NA might be expected to report this information back to its client either regularly or upon request. Further, NA might have to wait for the authorization of its client before committing to a certain decision. In the process of negotiating, NA may be mandated to contact its client under certain conditions. Furthermore, NA has the obligation of keeping its client aware of the latest shifts in the conflict just in case they may affect its views and requirements. At the occurrence of such changes, NA must reevaluate its position and its strategy for addressing this situation and hence, modify it if necessary.

The event trace of the negotiation's conclusion is illustrated in Figure 22. NA5 presents the resulting settlement to its client [e19]. It recapitulates to A5 the major decisions it took in the negotiation process. NA5 submits the list of commitments it has taken on behalf of A5. It also describes the constraints it has met. At this stage, NA5 returns control of both the session and the connection with A5 to NAG. NA5 also submits its log to NAG as part of its briefing on the session it conducted [e20].

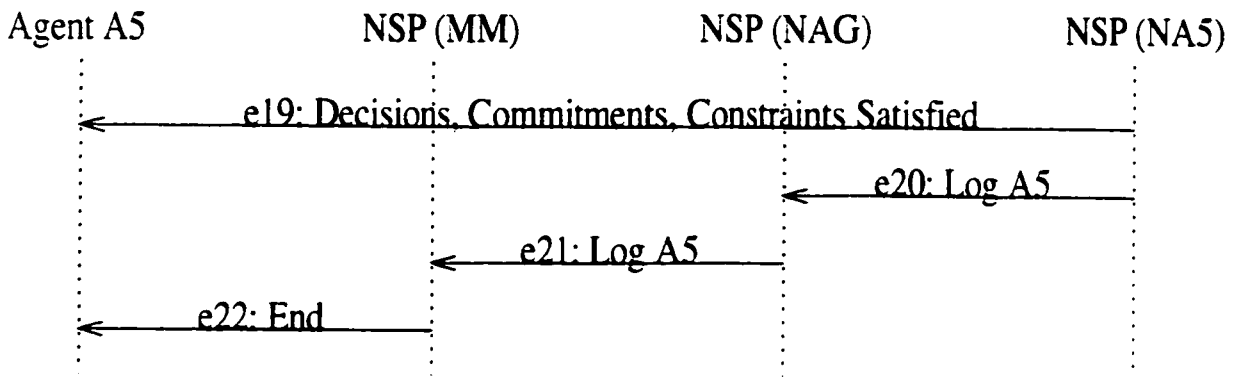

Figure 22: Event Trace of the Negotiation's Conclusion.

NAG communicates all this information to MM along with the control of the session and the connection to A5 [e21]. MM records all the information in its permanent logs. It then closes the record of this session and takes care of any remaining managerial issues. With these activities, MM concludes the session [e22].

Chapter 4

Conflict Description

Negotiation is a response to conflict as pointed out by Greenhalgh and Chapman [GC95]. The ultimate goal of negotiation is to resolve conflicts. In an agent's society, a large variety of tasks and objectives are pursued by different agents and conflicts arise. NSP provides these agents with negotiation services. The agent in conflict must know the context and the cause of the conflict it is facing when resorting to negotiation. To reach a settlement, the agent must be willing to make concessions and/or is capable of altering the other party's position in the conflict. An agent may make concessions if it has its own prioritized list of goals, its constraints and its margin of maneuver. In some situations, the agent may have acquired or learned some information that could possibly influence the other party involved in the conflict.

To illustrate these characteristics in describing a conflict, consider the example of the personal travel agent (PTA) of a company ACME organizing the conference trip of Joe Blow to London. The requirements for this trip may be divided into:

- Goals:
  G1: Travel from Montreal to London to attend a conference
  G2: Stop in Paris to visit friends

- Constraints:
  C1: The time period is the first week of September
  C2: Stay in a hotel in London
  C3: Total cost [Hotel + Registration + Transportation] < 2000$

- Implicit Constraints:

I1: For the period of the conference, Joe must be in London.

I2: Hotels near the location of the conference are preferred.

In the process of arranging this trip, the PTA contracts a travel agent (TA) and a hotel agent (HA). The TA is assigned the task of shopping for a plane ticket to London with a stop in Paris. This assignment is represented by the arrow originating from PTA's timeline and intersecting the TA's timeline in Figure 23. The PTA contracts the HA to look for a hotel within the specifications attached to the arrow. The arrow originating from the PTA's timeline and ending on HA's drawn in Figure 23 corresponds to this action. The annotations associated with arrows specify the goals and constraints corresponding to the sub problems delegated.

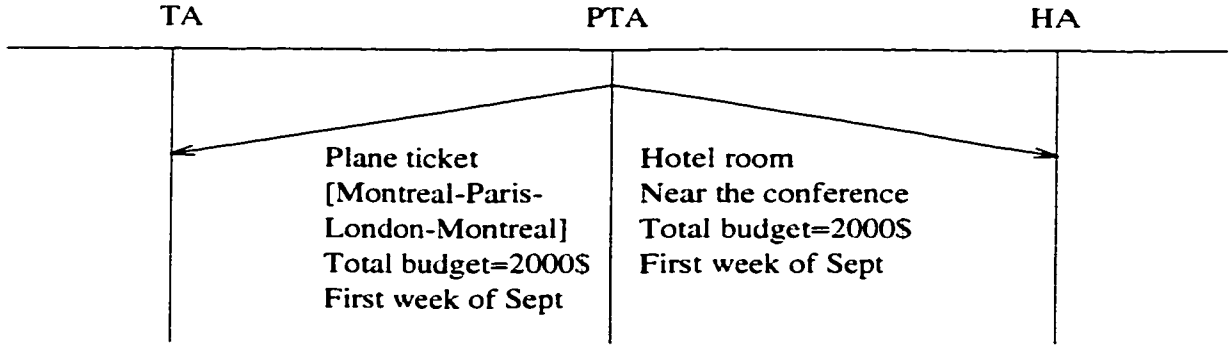

Figure 23: Conflict Tracing - Task distribution

This example problem will be used to describe the different aspects of conflicts presented in this chapter.

4.1 Context of the Conflict

In the process of performing tasks and pursuing goals, an agent faces a conflict. To seek negotiation, the agent must be able to recognize this conflict and to identify the tasks blocked due to this conflict. In the case of agents that have no capability to detect conflicts, it is possible that the agent they report to be the one that detects the contention on their behalf and identifies the tasks and goals in conflict. In either case, these blocked tasks and goals constitute the context of the conflict.

Detection of conflicts is not discussed in this thesis. An agent in conflict may contact a NSP to arrange an agreement with the other parties involved in the conflict.

The agent provides the context of its conflict to NSP as part of the required pre-negotiation information.

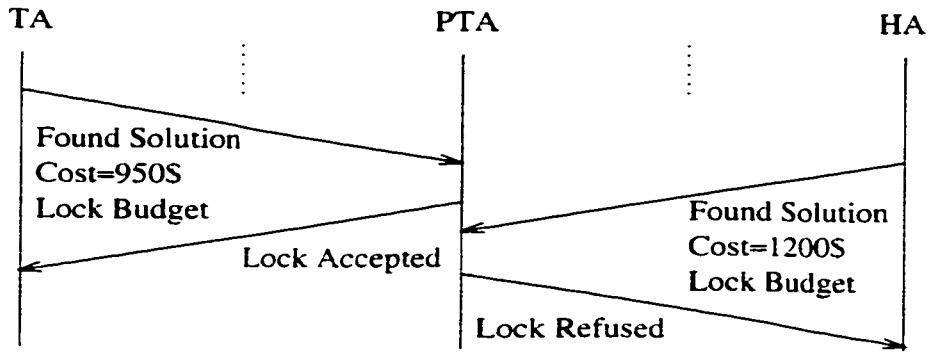

Figure 24: Conflict Tracing - Identification of the Context

Continuing the scenario mentioned at the beginning of this chapter, TA accomplishes its task and finds a plane ticket to London via Paris for 950$. TA informs the PTA to lock this amount in the budget with the message "Cost=950$ Lock Budget", as illustrated in Figure 24 by the directed line starting with the TA and finishing with the PTA. The PTA accepts the transaction and notifies the TA by sending a "Lock Accepted" message. TA's goal is reached upon the reception of PTA's acceptance of the TA's proposed solution. Now, only 1050$ are still available in the budget. Similarly, HA meets the requirements and finds a room for the duration of the conference. The total cost of renting this room is 1200$. HA notifies the PTA of its results and requests the allocation of this amount out of the budget to meet its goal. The HA sends the PTA the message "Cost=1200$ Lock Budget" as illustrated in Figure 24. The available budget no longer covers the expense of the room. The PTA replies to the HA proposal with a "Lock Refused" message.

The failure of HA to close the deal with the PTA although it reached a solution within the original specifications identifies a conflict between the two agents. HA's action to lock the price of the hotel room in London in the budget represents the context of this conflict. This context must be provided to NSP when requesting its negotiation services.

4.2 Cause of the Conflict

Along with its context, the cause of the conflict completes the initial pre-negotiation information required by NSP. The cause can be expressed as a combination of the identity of the conflicting agent and that agent's task or goal forcing it to refuse the request. It is not always possible for the agent facing a conflict to isolate the reasons for the other agent's rejection.

Depending on its sophistication, an agent will identify the cause of the conflict at different levels of granularity. The granularity level at which conflict is identified depends on the sophistication of the agent. In the simplest case, the agent may provide the identity of that agent denying it a service. Identifying which of the agents caused its uncooperative position becomes the NSP's responsibility.

We organized the sources of conflicts as illustrated in Figure 25.

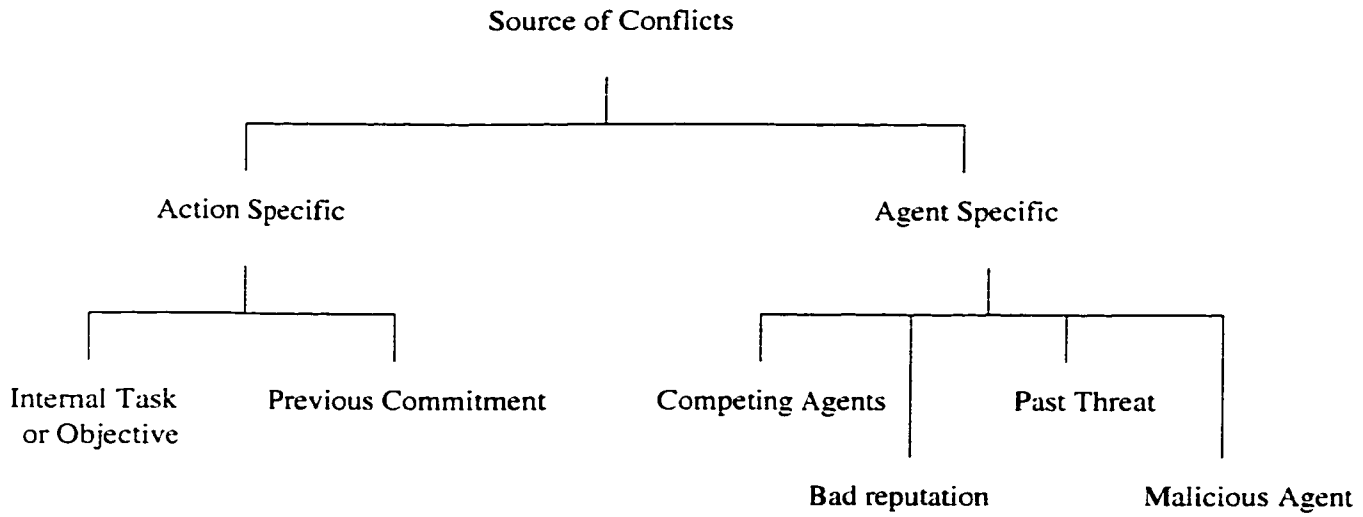

Figure 25: Source of Conflicts Taxonomy

Solving action specific conflicts involve modifying an action: the one requested, or the one that it conflicts with. An agent may refuse to cooperate when requested to perform an action having a negative impact on its own internal tasks or objectives. Thus, it is able to directly reveal its position to the requesting agent. An agent P may be unable to meet the needs of the requesting agent R because it has previously committed the needed resources to another agent C. Hence, it is up to the requesting agent R and agent C to collaborate in redistributing among each other the resources, the load or the tasks of agent P.

Suppose agent A is in conflict with a specific agent B and chooses not to cooperate. Although, agent A may be capable of performing an action Aj, it will refuse to perform it on behalf of B. Such agent specific conflicts are harder to solve and require convincing the conflicting agent A to ignore its past experience with agent B and take a risk by cooperating. Different reasons may promote such an attitude.

- Competing agents: The two agents may belong to different competitors.
- Bad reputation: The requesting agent did not keep its past commitments.
- Past threat: The requesting agent has been classified as a malicious agent.
- Malicious agent: The conflicting agent is executing a past threat ignored by the requesting agent.

Regardless of what the granularity level supported by the agent is, it is NSP's responsibility to gather relevant information to supplement what is provided by the agent. Based on the input from agents, it investigates the situation and builds the dependency trees between agents and tasks as well as among the tasks themselves. Figure 26 provides a collection of the dependency trees we propose. Four symbols: an X, an underlined $\underline{X}$, a circle and a square.

- X: An X represents a task or goal committed.
- $\underline{X}$: An underlined $\underline{X}$ represents the action facing a conflict.
- Circle: A circle represents a choice. For example, an agent has committed to a specific request but is willing to modify its position if the agent with that specific request accepts to alter it.
- Square: A square denotes an agent with an agent specific conflict.

X and circles are labeled with the name of the agent they belong to and the task or goal they represent. As for squares, they are labeled with the agent they belong to and the type of agent specific conflict they have.

Tree 1 in Figure 26 represents the case where the agent A2 refused the action T3 of agent Ai because it directly conflicts one of its own goals or tasks Tj. Tree 2 illustrates the case where the tree has a depth. This occurs when interdependent

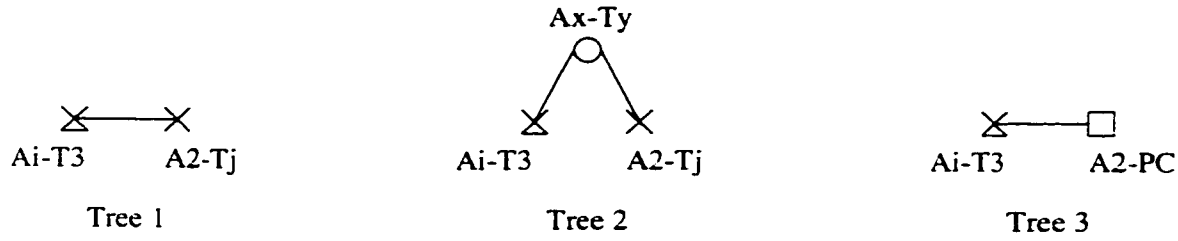

Figure 26: Dependency Tree Samples

tasks are involved in the conflict. In this case, agent Ax refused agent Ai's action T3 because it conflicts with action Ty. Ax committed to Ty upon the request of agent A2 for action Tj. Tree 3 depicts the case where agent Ai has encountered an agent specific conflict when it requested action T3 from agent A2. A2 is refusing T3 because of its experience from the past commitments and experience with Ai. Using these tree building blocks, NSP can reconstitute the dependency tree behind a conflict and isolate the agents and the tasks causing it. Once NSP has modeled the conflict and isolated the different items involved, it is able to initiate negotiations with the appropriate agents.

Based on the conflict presented in the previous section, HA must provide NSP with the cause of the conflict. HA receives the "Lock Refused" message. It contacts NSP. Among the information it provides to the NSP, HA includes the cause of the conflict. In this scenario, the immediate cause of the conflict is basically the PTA, since it is refusing the action requested by HA. NSP's initial task is to complete the data it has received from HA before launching the negotiation. It attempts to build the dependency tree of this conflict as depicted in Figure 27. It starts by contacting the PTA and requests its reasoning which lead to its lock refusal. The PTA explains that although the budget is 2000$, it has committed 950$ for TA reducing the maximum size of lock it can still allocate is 1050$. Therefore, it can not authorize a lock for 1200$. NSP pursues its goal of building the hierarchical structure of this conflict and its roots. NSP contacts the TA to extract the rationale behind its act. In its reply, the TA explains that its conflicting lock is for only 950$ from a 2000$ budget for the plane ticket requested by its client.

Now NSP has the complete picture:

- HA and its task Lock Budget for 1200$
- PTA and its previous commitment to Lock Budget 950$

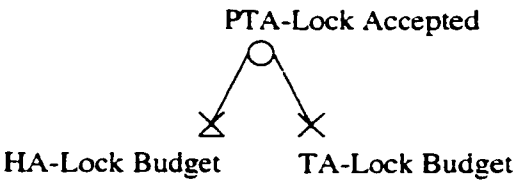

Figure 27: Dependency Tree for the Hotel Agent conflict.

- TA and its task Lock Budget for 950$

Hence, NSP knows the parties to negotiate with and they are: HA, TA and PTA.

4.3 Agent's Own Prioritized List of Goals

An agent typically achieves at least one ultimate goal during its life cycle. We broadly classify goals into two categories:

- Atomic: Atomic goals are goals that are simple and conveniently manageable as they are. Subdividing atomic goals has no added value.
- Compound: Compound goals are divisible into subgoals iteratively until atomic goals are reached.

The progress of a complex goal may be measured with the number of its atomic goals that has been reached. In networked and distributed environments, agents with compound goals typically contract other agents for specific subgoals. Therefore, more than one agent may be collaborating to achieve a common goal. Depending on the nature of the activities involved in reaching this goal, the cooperation among those agents could vary from requiring a tight and close contact to maintaining a loose and highly independent relationship. Thus, it becomes highly probable that two agents are in conflict without being aware that they both are working to achieve the same common goal at a higher level of abstraction.

In the context of multiple goals, an agent prioritizes these goals to organize its tasks and resources. Goals are handled based on their priorities unless they have similar priorities in which case they are addressed in an arbitrary order. These prioritized lists of goals can provide crucial means to resolve discords among conflicting agents.

NSP would typically consult the prioritized list of each agent involved in the conflict. When these lists include goals inherited from other agents, NSP requests the prioritized lists of those other agents to reconstruct the global prioritized list of goals. If it finds a common goal among the conflicting parties, NSP uses that goal as a milestone in negotiating an agreement. In such circumstances, usually, an action reaching the immediate subgoal of its agent is effectively hindering the achievement of the ultimate goal which is at the base of its own subgoal. With such information, NSP has a better chance of convincing the parties with such conflicting actions to retract them.

Within the context of the conference trip example mentioned in this chapter, NSP is "currently" aware of the context of the conflict as well as its cause. NSP decides to consult the prioritized lists of goal for each of the three parties involved in the conflict — HA, PTA and TA. Based on these lists, it builds the global prioritized list of goals and looks for a common ground to reach a settlement.

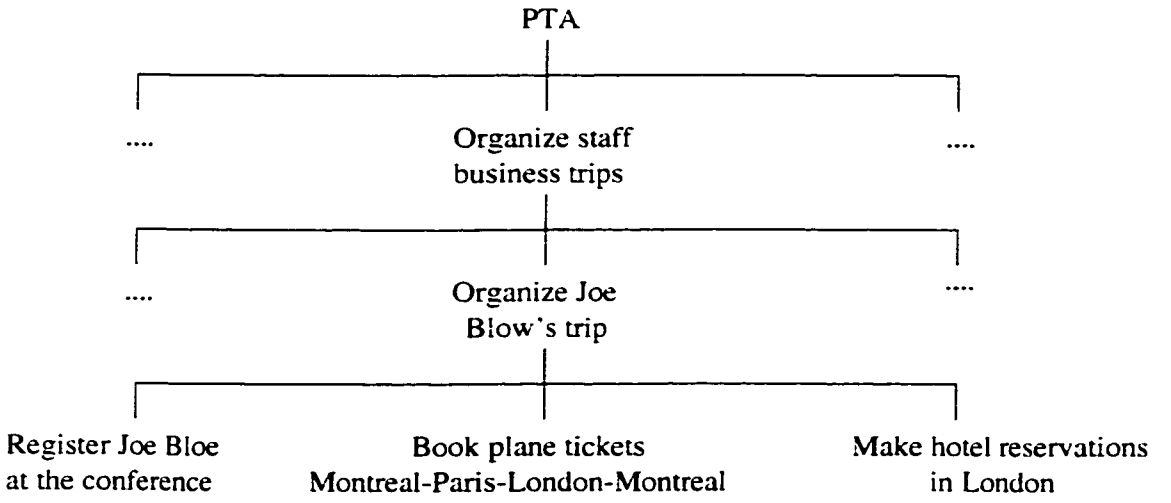

Figure 28: Partial Prioritized List of Goals of PTA

Let us consider the prioritized list of goals for the PTA traced in Figure 28. The PTA organizes business trips of ACME's staff. Specifically, it is organizing Joe Blow's trip. Therefore, the PTA has to:

- Register him at the conference he is attending in London.
- Book the plane tickets to London via Paris for Joe Blow's trip.
- Make the hotel reservations in London for Joe Blow's trip.

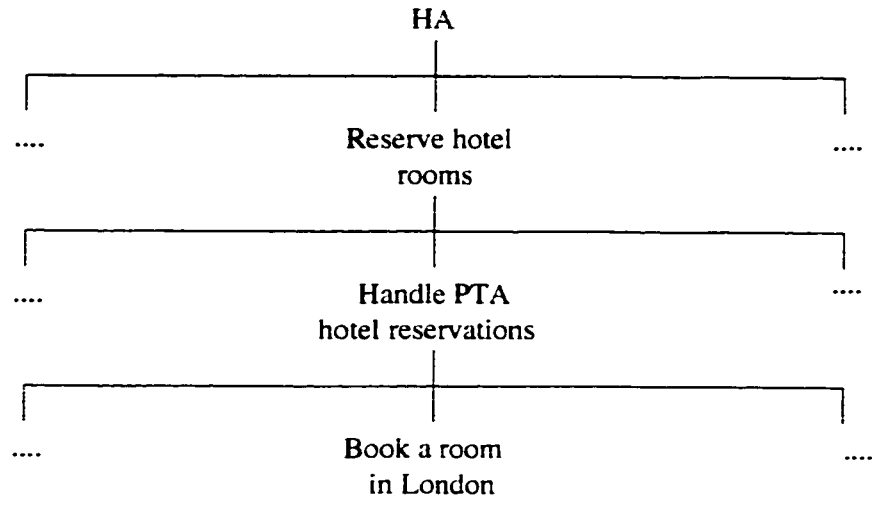

Figure 29: Partial Prioritized List of Goals of HA

The prioritized list of HA's goals is depicted in Figure 29. The ultimate goal of HA is to reserve hotel rooms. In this case, it is handling the PTA's hotel reservation. It has to book a room in London according to PTA's specifications.

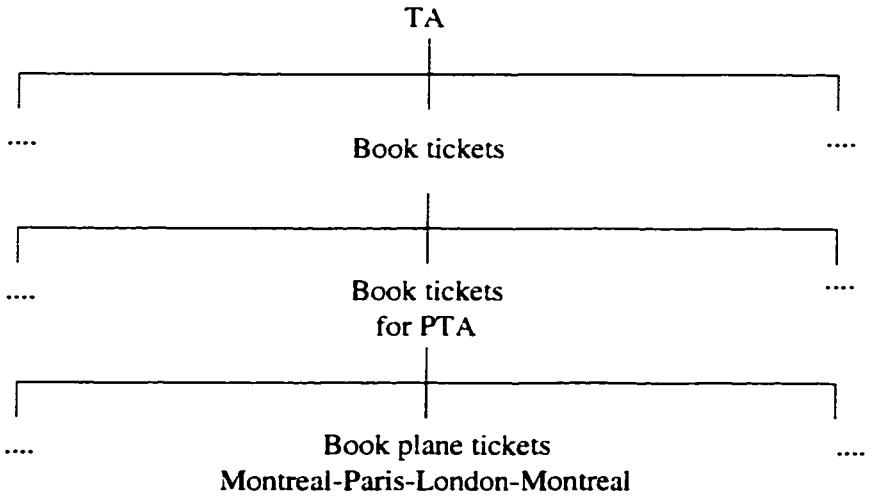

Figure 30: Partial Prioritized List of Goals of TA

Figure 30 presents the TA's prioritized list of goals. TA's goal is to book tickets. In particular, it is booking tickets for the PTA. The plane ticket is from Montreal to London via Paris.

From the prioritized list of HA, NSP discovers that HA's need to lock the budget with PTA results from HA's goal to book a room in London. Traversing the list shows that this goal is a subgoal of "Handle PTA hotel reservations". Investigating TA's

prioritized list, NSP learns that the task of locking the budget is part of the TA's goal to book plane tickets Montreal-Paris-London-Montreal. Going one step higher in the hierarchy, TA's goal is to "Book tickets for PTA". Moving to the prioritized list of PTA, NSP discovers that PTA's specifications provided to HA are for its goal "Make the hotel reservations". PTA's specifications for TA are for its goal "Book the plane tickets". These two goals are effectively subgoals for "Organize Joe Blow's trip". This global prioritized list of goals built by NSP is depicted in Figure 31. This list shows that both TA and HA are working on two sub divisions of the same problem of PTA, namely organizing Joe Blow's conference trip to London. Using this information, NSP is able to convince

- PTA and TA to reconsider their commitment to lock the budget.
- HA, PTA and HA to cooperate in distributing the budget among them

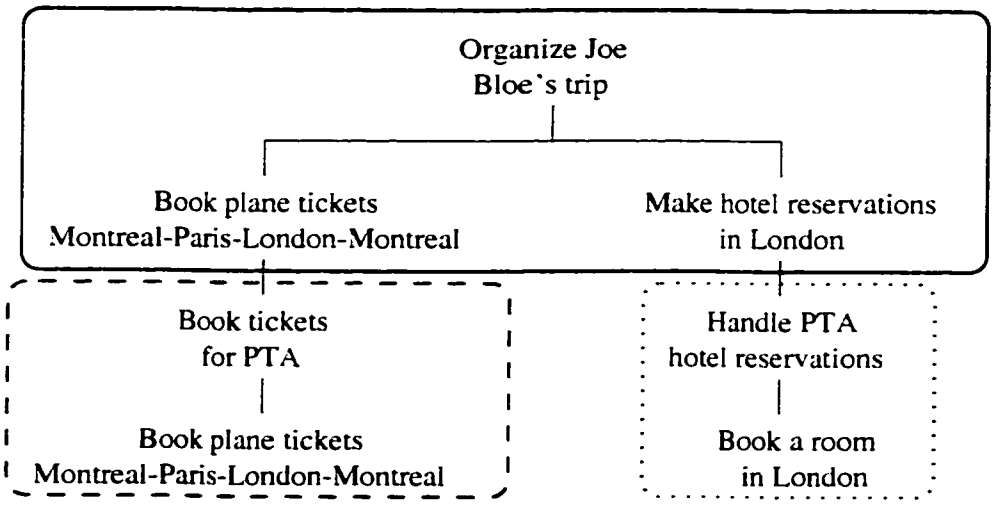

Figure 31: Global Prioritized List of Goals

An agent maintains its own priority list has higher potentials of achieving its goals and thus successfully operating.

4.4 Constraints

Constraints play an instrumental role in directing the agents towards an acceptable solution. Depending on the nature of the task or objective and the context, different

sets of constraints apply. These sets are made of various combinations of issues such as price, quality, deadlines, capacity and timings, etc. Every detail specified in the process of identifying the task or objective can take the role of a constraint.

The grammar used in formulating constraints is based on the structure of the constraints' expressions introduced in Figure 32. To describe this grammar, we use the following typographical conventions. Table 10 shows the grammar.

- Non-terminal symbols are written all in lower case. Examples are: *expression*, *selement*(simple element), *celement*(complex element).
- Terminal symbols have an initial upper case letter. Punctuation symbols are quoted. Examples are: *Number*, "}".
- The following symbols are meta symbols of the grammar:
  - $\rightarrow$ links the defined symbol to its defined expression
  - {} are used to group expressions

expression	$\rightarrow$	"{" *expression* "}"
	$\rightarrow$	"{" *expression* "&" *expression* "}"
	$\rightarrow$	"{" *expression* "|" *expression* "}"
	$\rightarrow$	*selement* | *celement*
selement	$\rightarrow$	*discrete*
	$\rightarrow$	"[" *discrete* ".." *discrete* "]"
celement	$\rightarrow$	"{" *celement* "," *celement* "}"
	$\rightarrow$	"(" *condition* "," *selement* ")"
condition	$\rightarrow$	*discrete* " =" *discrete*
	$\rightarrow$	*discrete* " <" *discrete*
	$\rightarrow$	*discrete* " >" *discrete*
	$\rightarrow$	*discrete* "! =" *discrete*
discrete	$\rightarrow$	*Number*|*String*

Table 10: Constraint Expressions Grammar

We represent each constraint by the following pair:

$$< name, value >$$

name is the name identifying the constraint and *value* formulates the actual value assigned to this constraint. The elements of the pair are separated by commas , and delimited with greater than and less than signs <>.

We associate the *value* of a constraint with two properties: requirements and category of constraint. The requirement part indicates the single or multiple alternatives available to satisfy the constraint. The category part of the constraints are divided into three types: discrete, interval based, conditional. As depicted in Figure 32, each constraint definition is based on the combination of a requirement type and a category type.

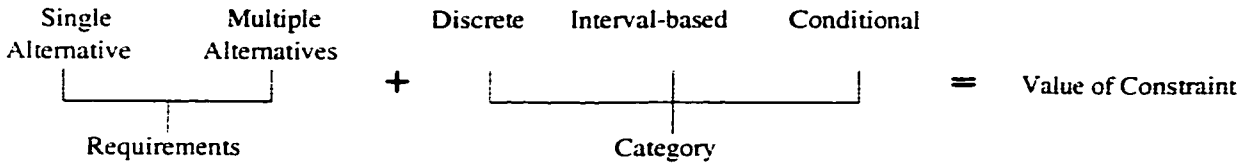

Figure 32: Constraint Specification

The simplest way of defining a constraint is by attributing to it a simple single discrete alternative. The original budget constraint of our example illustrates this type of definition and is assigned a simple single value, namely 2000$. Based on this grammar, the syntax for representing such type of constraint is:

$$< budget, 2000\$ >$$

The value of a constraint may be compounded in the case of discrete values. The initial destination city constraint is a good illustration. The initial itinerary requires stops in both London and Paris. This destination is compounded because it is based on two distinct cities where each city — London, Paris — is a valid destination as it stands. Further one city cannot be replaced by the other one in the itinerary. This destination is also a single alternative because there are no other acceptable destinations to replace the compounded destination of London and Paris. The compounded values are isolated by ampersands & gathered between curly brackets {}.

$$< destination\ city, \{London\&Paris\} >$$

If we assume Joe Blow's conference is scheduled in two cities, then London alone is not an alternative. If the conference were to be held in London, Oslo and Tokyo, Joe can participate in this conference at any of its locations. This scenario illustrates the simple multiple discrete alternatives type of constraints. This example is considered simple because each alternative is one distinct city and cannot be subdivided further.

The alternatives are separated by pipes | symbols and enclosed between two curly brackets {}.

$$< destination\ city, \{London|Oslo|Tokyo\} >$$

Suppose, regardless of in which city Joe Blow attends the conference, he wants to stop in Paris to visit his friends. Then the scenario illustrates the compound multiple alternatives type of constraints. Because a destination combines two cities, this is a compounded constraint. Following the grammar rules presented in table 10, this constraint is expressed as follows:

$$< destination\ city, \{\{London\&Paris\}|\{Oslo\&Paris\}|\{Tokyo\&Paris\}\} >$$

or:

$$< destination\ city, \{\{London|Oslo|Tokyo\}\&Paris\} >$$

The examples mentioned so far hold discrete values. However, sometimes an interval is a more appropriate representation for the constraint. For example, some travelers prefer arriving at a city they are not familiar with during the day. So any flight arriving to New York between 7:00 and 18:00 is acceptable. This scenario illustrates a simple single alternative interval-based constraint. The boundaries of the interval are isolated by two dots .. enclosed between square brackets [].

$$< arrival\ time, [7:00..18:00] >$$

A compound multiple alternative interval-based constraint is shown below. This expression represents the schedule Joe Blow desires at the conference for him to attend the session. They have to be within these allocated time periods. They should all be in the early morning and late in the afternoon or between late morning and early afternoon.

$$< conference\ time, \{\{[8:00..10:30]\&[16:30..19:00]\}|[11:00..16:00]\} >$$

In some situations, a set of tuples formed by a condition and a value or an interval illustrates the situation more adequately. Consider this example. In London, Joe Blow usually stays at the Ritz Carlton. In Paris, he prefers staying at the Hilton or Hotel Napoleon rather than the Ritz. As for Montreal, his favorite is the Queen Elizabeth or Vogue. This constraint is expressed below following the grammar introduced in table 10.

$$< hotel, \quad \{(city = London, RitzCarlton),$$
$$(city = Paris, \{Hilton \mid Hotel\ Napoleon\}),$$
$$(city = Montreal, \{Queen\ Elizabeth \mid Vogue\})\} >$$

The example introduced at the beginning of this chapter carries its own set of constraints. The budget of 2000$ represents the financial constraint. The destination cities: London and Paris are also considered to be constraints. The location of the conference is a constraint in the search for a hotel for Joe Blow. The time constraint in this example has two aspects. These reservations and bookings are for the first week of September. Since Joe works for a company, let us suppose that he can't stay away from the office longer than what is absolutely required. This time constraint is also reflected on the transportation constraint. Hence, flying is the only accepted means of locomotion.

$$< budget, 2000\$ >$$
$$< destination\ city, \{London\&Paris\} >$$
$$< hotel\ location, London >$$
$$< trip\ duration, [Sep\ 1..Sep\ 8] >$$
$$< locomotion, Plane >$$

With a better grasp of the requirements expressed through a set of constraints specification, the search problem becomes a constraint satisfaction problem. There are three factors involved in specifying constraints: the number of constraints, the granularity of the details in the constraints and the cost involved in both introducing new constraints and specifying their requirements.

The less constraints are defined, the more options are available, making the search space larger. On the other hand, the more constraints are imposed, the more the search is limited thus reducing the alternatives to a relatively smaller set.

Further, specifying constraints at very low granularity levels unnecessarily introduces complexity and may potentially hinder producing a matching solution. Whereas with high granularity, some crucial details are ignored and the better matching solution can be missed.

The cost of defining constraints is a function of the number of constraints already specified and the granularity of their requirements. The more constraints are introduced and the more detailed their requirements are, the more resources are used in

specifying them. Further, the more complex the problem description is, the more expensive it is to find good matches.

Specifying the optimal set of constraints and their requirements is highly application and context dependent. Building this set can be quite a complex task. In some traveling applications, a constraint such as the destination city can be crucial whereas in some other applications, the destination city may not be that critical. A constraint such as the time takes up different dimensions depending on the application. The grammar presented in this section is not all encompassing. Specialized applications may need to expand the grammar before using it. Negotiation agents may have this grammar built in. When they meet conflicts requiring finer granularities in specifying constraints, they could request the more advanced grammar from their NSP. Despite the difficulties inherent in constraint specifications, building the set of constraints and their respective requirements form a good start in the process of settling a conflict.

4.5 Margin of Maneuver

Among all the constraints specified within a problem, some are more crucial than others. Further, among the different potential requirements of the same constraint, some alternatives are preferred than others. The margin of maneuver of a constraint is a hybrid measure of both its degree of relevance within the problem at hand and the nature of its requirements.

The relevance of a constraint determines to which extent a potential solution must meet this constraint. A constraint may vary from being essential — must be fulfilled — in one extreme to being simply optional — could be ignored — at the other end.

The nature of the constraint's requirements between the two extremes determines the degree of flexibility in conforming to the constraint's satisfaction. Figure 33 depicts this spectrum. The harder the constraints are, the lower is their margin of maneuver and vice versa.

Hard constraints are at the core of any potential solution. Joe Blow's conference trip had a few hard constraints. The destination city is a constraint at the core of the problem. A trip without a destination city cannot even be considered as a solution. Further, a destination other than London is not accepted since the purpose of the trip is to attend the conference taking place in London only. Flexible constraints

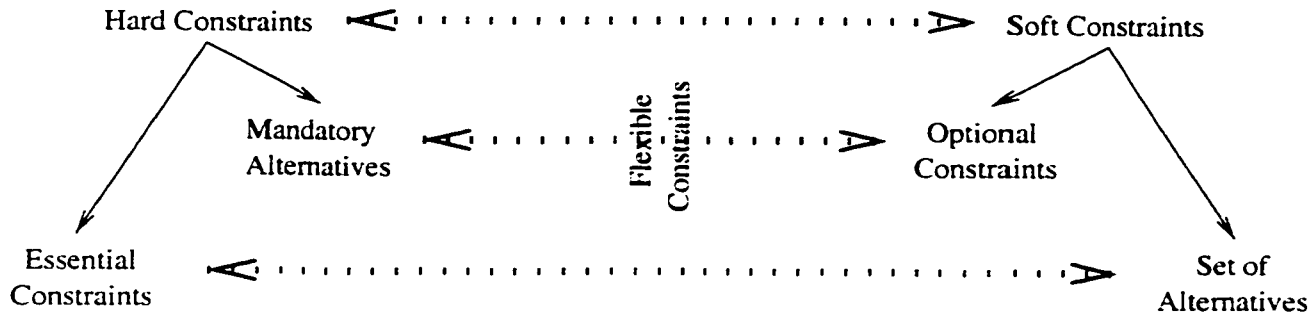

Figure 33: Margin of Maneuver Spectrum.

are requirements highly desired to be satisfied. A solution lacking some of these constraints could be treated as a partial solution. The trip duration is an example of a flexible constraint. If there are no flights arriving at London on September 1st, arriving on August 30th becomes a viable second choice. Soft constraints are typically preferences. The solution satisfying them gains some added-value for it. Soft constraints play a major role in selecting a solution from the set of available solutions. A window seat away from the wings can be listed as a constraint but a solution with a window seat could be considered acceptable.

A continuum links the three distinct categories of this example as illustrated in Figure 34. A hard-flexible constraint is the example where the conference is held in three different cities: London, Oslo and Tokyo. Joe Blow has to attend this conference making the destination a hard constraint.

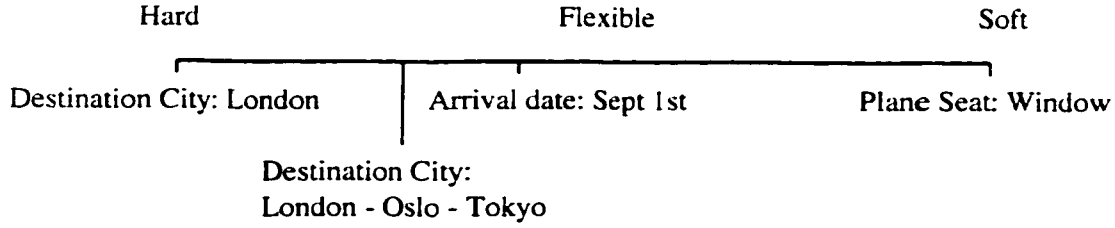

Figure 34: Margin of Maneuver Classification of Some Examples.

Knowing the margin of maneuver of the constraints is instrumental in finding a near-optimal solution satisfying the set of constraints. This information could be embedded in the constraint specifications by means of weights. We associate weights both at the level of the constraint itself and at the level of the requirements. Weights are assigned in the range from 0 to 1. A weight is attached by appending a column : then the value of the weight to the entity weighted. The syntax for specifying weights

is as below. The *budget* constraint has been assigned a weight of *1*, whereas its single requirement, *2000$*, has been assigned the weight *0.98*.

$$< budget : 1, 2000\$: 0.98 >$$

A weight of 1 identifies a hard constraint or requirement. Therefore, it must be satisfied. Any weight less than 1 identifies a flexible constraint which could be relaxed if deemed necessary. The closer the weight is to zero, the less the constraint is required and hence it may potentially be ignored.

Along with the constraint specification, the margin of maneuver plays an instrumental role in solving problems where the boundaries separating an acceptable option from an unacceptable one are fuzzy. In the process of negotiating a solution for the problem at hand, a solution satisfying all the specified constraints is the initial target. In some cases, such a solution might be impossible to achieve. Then, the agents identify the set of constraints causing the conflict and gradually relax the ones with the lowest weight to reach what could be qualified as a near-optimal partial solution.

In fact, a closer look at the constraints in Joe Blow's conference trip shows that some constraints are more flexible than others. Although the initial search for a plane ticket included the two destinations, a careful look at the constraints shows that visiting Paris is a flexible constraint whereas London is a hard constraint. This information can be expressed as follows:

$$< destination\ city : 1, \{London : 1 \& Paris : 0.6\} >$$

Constraints with weights provide a more flexible means in finding acceptable solutions. When a conflict arises in satisfying hard constraints for Joe Blow's trip, constraints and requirements with high level of maneuver like Paris may be relaxed to meet the harder constraints. Consider the conflict between satisfying the budget limitations and covering all the desired destination cities specified earlier in this section. If the margin of maneuver of this trip's constraints are specified, NSP would be able to negotiate a refined compromise. Without the extra information, solicited through weights in our proposal, NSP may produce a less optimal compromise or find no feasible solution at all.

4.6 Information

Negotiation is not restricted to making concessions. It also involves argumentation, concluding deals and issuing threats. These types of negotiation tactics cannot be used casually. For effective results, the agent must be aware of some preliminary information regarding the other agent(s) involved in the conflict and the situations which could influence their actions. Alternatively, it must have some "deep insight" of the situation in order to forge an agreement.

Argumentation is possible among agents of varying authority and power. It gives the best results for agents with limited resources having conflicts with more influential agents. Consider the following two conference scenarios in Joe Blow's company:

- Typical Conference Scenario:
  Budget for Conferences = 500$
  Duration = 1 day
  Location = North America

- Joe Blow's Conference Scenario:
  Budget for Conferences = 500$
  Duration = 1 week
  Location = London

The PTA allocates a budget of 500$ to both TA and HA. TA returns with the following deals:

- **T1:** Destination = {London & Paris} : Cost = 550$

- **T2:** Destination = {London} ; Cost = 400$

HA proposes the following two deals. The second choice is the only one matching the company's standards.

- **H1:** Distance from conference = {10Km} ; Rating = {3 stars} ; Duration = {1 week} ; Cost = 50$

- **H2:** Distance from conference = {5Km} ; Rating = {4 stars} ; Duration = {1 week} ; Cost = 80$

In selecting from the deals produced by the agents, the PTA must respect the following formula:

$$\text{Cost of Plane ticket} + \text{Cost of a hotel room} \times \text{Duration} \leqslant \text{Budget}$$

Even by accepting the cheapest deals by both the travel and the hotel agents (T2 & H1), the PTA exceeds its allocated budget as shown below:

$$400\$ + 50\$ \times 7 = 750\$ > 500\$$$

The PTA contacts the NSP for help in resolving this conflict. The NSP attempts to both reduce the costs and relax the budget constraint. Therefore, it consults the archives of Joe's company to compare the current costs of the conference to the ones it paid in similar circumstances. It discovers that the latest plane ticket bought for London cost 450$ and a room cost 90$ a night. Further, Joe's conference lasts a week and is held on another continent.

With these results, NSP realizes that the budget is overconstrained and possibly inadequate for this trip especially since both agents were not able to reach a solution within all the constraints specified. Given that the company has a precedence in authorizing such expenditures, NSP decides to contact the budget manager agent and request an increase in the budget. The following are its supportive arguments:

- Current cheapest hotel room costs 50$ a night versus 90$ a night paid by the company in the past
- Cheapest plane ticket costs 400$ versus 450$ paid by the company
- The 500$ budget is based on a one day conference in North America whereas this trip is a week long and is held in London.

The budget manager verifies the information provided by NSP in its disbursement history. Because the increase is justified, the manager reevaluates the budget and raises it to 2000$. Using argumentation, the NSP was able relax the constraint causing the conflict.

With agents of equal influence and control, "making deals" is the best strategy for an agent to achieve its objectives when they conflict with other agents. Deals can

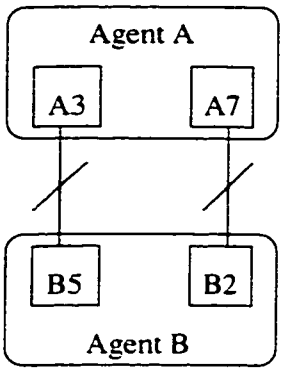

Figure 35: Agents Making Deals.

either be short term or long term. With short term deals, the benefits are immediately experienced. Consider the scenario depicted in Figure 35.

Agent A has plans A3 and A7. Agent B has plans B2 and B5. Plans A3 and B5 are in conflicts. Plans A7 and B2 are also in conflict. In such a scenario, NSP is able to negotiate that each agent drops one plan in favor of the other one. Agent A may accept to abort its plan A3 if agent B accepts to abort its plan B2. Thus agent A can pursue A7 and agent B can continue B5 which have no conflicts.

The benefits of long term deals are typically realized in the future. The agent willingly accepts to contradict its objective and the positive implications of its action are experienced at a later stage. Consider the same scenario presented to illustrate the argumentation tactic. In its attempt to reduce the costs, NSP makes a deal on behalf of TA with Air Canada's agent.

- Air Canada's Agent: Reduce the ticket cost to 400$
- TA: Commit to send a minimum of 40% of its customers per month on Air Canada

Consulting its logs for the past six month, TA has been meeting the requirements of this deal on regular basis and will get an extra discount for that. Therefore, TA accepts this deal. Air Canada's agent reduced its immediate profit with the expectation of guarantying a minimum amount of sales per month. Thus, Air Canada's agent concludes a long term deal.

Some agents might resort to threats as a way to solve their conflicts. A very powerful agent may effectively threaten not-so-powerful (simple) agents and force

them to abort their objectives in order to achieve its own intended goal. Similarly, a simple agent which happens to be able to sabotage critical objectives of some influential agents may get its way in exchange for not interfering with these objectives. Assume Joe Blow's company has an exclusive agreement with a travel agent TA1. All the business plane tickets paid by the company are arranged through TA1. Because of the new cuts in this year's budget, additional discounts on tickets are mandatory. NSP issues the following threat to TA1:

- If a 10% discount is not applied on TA1 prices, the PTA will breach the exclusive contract.

Since Joe's company is its major customer, TA1 cannot afford such a loss. So TA1 finds itself forced to reduce its profits and abide to this ultimatum in order to stay in business.

4.7 Conclusion

The six different areas we identified in this chapter describe a conflict. In the process of presenting these areas, we introduced new ideas such as:

- A classification the sources or causes of conflicts
- Weights to quantify the margin of maneuver
- Dependency trees to build the agent's prioritized list of goals
- A grammar to express constraints

Information regarding these areas of conflict description are collected and used by NA. Constraints are used to select the appropriate negotiation protocol to use. The remainder of the areas guide NA in the decision making process involved in resolving a conflict.

Chapter 5

Characteristics of Different Negotiation Protocols

The purpose of NSP is neither to introduce new negotiation protocols nor to enhance known ones. NSP offers the necessary infrastructure to integrate old as well as new negotiation protocols and make them available to the agent community through a standard interface. In this chapter, we introduce three widely known protocols: the Contract Net Protocol, the Multistage Negotiation Protocol and the Partial Global Planning Protocol. Then we present a methodology to add a new negotiation protocol to the protocol suite in NSP. Given the constraints of a conflict, this methodology facilitates the selection of the appropriate protocol to use in the negotiation. We illustrate this methodology using the three widely known protocols.

5.1 Classical Protocols

5.1.1 Contract Net Protocol

The first protocol we introduce is the Contract Net Protocol developed by Smith and Davis [Smi98, DS83] for task and resource distribution among agents also known as nodes. The Contract Net is made of a collection of nodes. A node may assume two roles: a manager or a contractor. Manager nodes monitor the overall execution of tasks while dividing them into subtasks and then assigning these subtasks to contractors. Contractor nodes are responsible for the execution of the tasks they bid on and win. These roles are not mutually exclusive. A node may dynamically take on either

role. The bidding process is initiated by a manager announcing a task. Available contractors, in the net, evaluate the manager's proposal based on their own abilities and commitments. They submit their bids for the most suited tasks. The manager evaluates the received bids, selects a contractor and awards the contract to it. The manager then monitors the execution of the contract. Contractors have the possibility to further partition its task and initiate bidding processes. Hence, the contractor takes on the role of the manager when subcontracting follows.

The protocol is a one-time, two-way exchange of information and a mutual selection process. The Contract Net Protocol has made two major contributions to AI:

1. Dynamic task distribution in a loosely coupled environment. Agents may be dynamically introduced and then removed. Load balancing is inherent since busy agents do not bid.

2. Two-way communication, or negotiation. The concept of negotiation is introduced in its simplest and most primitive form. It is a one-shot attempt to negotiate. Nevertheless, it opens the door for the introduction of more sophisticated strategies of negotiation such as using counterproposals to find a common ground for both the contractors and the bidders.

Many researchers followed the initial work on the Contract Net. In the critique of the Contract Net, we mention the following limitations:

- This protocol does not detect conflicts and does not support bargaining between the agents. There is no provision for compromise or generation of alternative solutions. Nodes involved do not communicate the reasons and assumptions of their decision. The manager does not communicate its minimal condition, nor do the bidders have a second choice [CW94]. Hence, constraint relaxation is not supported.

- Agents are assumed to be benevolent and cooperative which is not always the case in reality.

- It is a rather communication-intensive protocol and involves multi-cast communication from manager to contractors. Its cost may outweigh some of its advantages in real-world applications.

- The protocol does not have any provision for report generation. However in an open system environment populated by heterogeneous agents belonging to different parties tracking of the compromises can be crucial.

- The protocol does not guarantee an optimal task distribution due to the cost induced by extensively transmitting information between contractors before awarding contracts.

5.1.2 Multistage Negotiation Protocol

The second protocol we present is the Multistage Negotiation Protocol developed by Conry et al. [CML98] as a generalization of the Contract Net Protocol. In the first phase of the protocol, agents generate "plans". Each agent locally instantiates a list of top level goals. For each goal, the agent builds a space of plans to satisfy it. The goals which are partially satisfied locally are called the primary goals or p-goals for agent i. Other agents may also contract agent i to help them satisfy their goals. These new goals become the secondary goals or s-goals of agent i. The next phase is the plan commitment phase. Each agent considers its set of p-goals and tentatively commits to the highest rated ones. Then, it contacts the other agents affecting its p-goals to confirm their commitment to these goals. All incoming requests for confirmation are handled by adding their s-goals to their set of active goals. Also responses to its own requests are incorporated into its feasibility tree, (AND-OR tree of the different goals, subgoals and plans). Using all the p-goals and the s-goals, a revised set of tentative commitments is made. New plan fragments are added and old ones removed. These changes are propagated to the other agents. The process of consulting incoming messages is restarted and plans are revised. This loop ends when the agent is aware of all the conflicts caused by its plan fragments.

The Multistage Negotiation Protocol replaces the one-shot question-answer model of the Contract Net protocol by a dialog or conversation. It allows iterative exchange of plans and their impact on other agents' plans before agents commit to an acceptable solution under these circumstances. It has made the following contributions to negotiation research:

1. It supports subgoal interactions in the context of distributed systems with no global views and no centralized control.

2. It permits the detection of overconstrained situations, that is situations where goals are not attainable.

3. It provides a mechanism for achieving consensus among agents facing conflicts such as resource allocation and compatibility conflicts.

Despite its effort to improve the Contract Net Protocol, this protocol exhibits the following limitations:

- It does not resolve or analyze all types of conflicts such as conflict of interest or cognitive conflict [CW94].
- The optimization issue is not addressed. The authors consider that the heuristics used ensure a fairly thorough search.
- The protocol does not have any provision for selecting the best choice in a situation where multiple alternative solutions are feasible.
- Although this protocol supports decentralized control, it lacks support for "open systems".
- The protocol does not use a "communication language" to facilitate knowledge sharing in cooperative problem solving. It doesn't even give a model for message exchange. It is up to the developer to select or device the syntax and the semantics used to communicate information.

5.1.3 Partial Global Planning

The third protocol we present here is the Partial Global Planning Protocol developed by Durfee and Lesser [DL98]. In this case, the nodes use a blackboard-based architecture for inter-agent communication. Each node forms its local plan. The node's planner combines its local plans into a node-plan based on the goals, order of activities and their estimated duration. Based on the node-plan, the planner generates the node-plan's activity map listing each activity with its predicted start-time and end-time as well as the result track. By comparing local plans to determine if they are part of a larger goal, the planner identifies partial-global-goals (PGG). For each PGG, the planner forms a partial-global-plan (PGP) holding the concurrent activities and intentions of all the nodes working on the same problem. The

planner builds a solution-construction-graph forming a high-level view of how nodes are pooling resources and working together. New PGPs, activity maps and solution-construction-graphs are generated iteratively until nodes converge on a distributed plan acceptable to all agents concerned.

Three different styles of cooperation can be used with this protocol. In one style, a central node generates and distributes the plans to the others. In another style, nodes work independently but synchronize their local solutions with others to achieve global solutions. In the last style, nodes negotiate and contract out tasks.

The Partial Global Planning protocol introduces a unified framework supporting different styles of cooperation. The contributions of this protocol may be summarized as follows:

1. With PGP, the solution time is substantially reduced at the cost of increased communication.

2. Without a global view of the problem, the protocol can still resolve conflicts by generating compromised PGP.

3. The protocol tolerates various levels of autocracy and democracy, obedience and insubordination within its nodes.

4. Nodes may change their meta-level organization.

Despite the originality of the negotiation technique in this framework, it has the following drawbacks:

- The negotiation is at a fixed level and the nodes may not communicate the rationale behind their decisions [CW94].

- "Misrepresentation or lying" and "exerting authority or threats" are the two suggested ways of achieving competing goals within this framework. Also Sycara et al, have argued that threat is not a viable solution and argumentation has a better outcome [KSE98].

- Basic negotiation techniques such as handling conflicting goals by arguing and comparing different solutions is not taken advantage of.

- The meta-level organization is assumed to be statically defined during network creation so it is not clear to which extent this protocol is valid in an unstable network where nodes are created and destroyed dynamically.

- The nature of the application might require different negotiation and cooperation techniques that are not necessarily based on the inability to communicate and the inconsistencies of the partial global plans. There is no provision in the protocol for such possibilities. The evaluation of the framework is based on the degrees of partial planning. Hence, the variation in the mechanisms is solely quantitative and not qualitative.

- Planning is not done optimally to reduce overhead.

- In an environment with nodes having low uncertainty on how to communicate, unnecessary overhead is introduced.

5.2 Protocol-Constraint Table

We propose a Negotiation Protocols Suite (PS) which is a combination of a set of protocols and the necessary methods to manipulate the data structures used to represent these protocols. It is at the core of NSP. One of the main objectives of PS is to find the appropriate negotiation protocol for its client. Therefore, in addition to providing its implementation, a protocol integrated to PS must be classified based on its characteristics.

The classification methodology we devised is based on the simple premise that the characteristics of a protocol may be defined in terms of the constraints in the context of a conflict. From the client of NSP, the negotiation agent (NA) spawned by NSP identifies a set of constraints C to be satisfied in the negotiation process. On the other hand, C is also an abstraction of the desirable properties of the protocol to be used. Therefore, PS combines protocols with constraints to construct a Protocol-Constraint Table (PCT). In PCT, each row characterizes a protocol. Each column represents a constraint. (i,j)-th entry represents the rating of the protocol "j" against the constraint "i". A skeleton of PCT is illustrated in table 11.

When a new protocol is added to the PS, a new row is added in table 11. The new protocol is rated against the already defined characteristics (columns) of the table.

Protocol Name	Function Name	Constraint 1	Constraint 2
Protocol 1	P1()	0	9
Protocol 2	P2()	10	5

Table 11: Skeleton of the Protocol-Constraint Table

The rating varies from 0 to 10. The increase reflects the higher support for this characteristic. Hence, 0 is attributed to an unsupported characteristic whereas 10 is assigned to a fully supported characteristic. Further, for each new characteristic added but not yet defined in the table, a new column is created. With the introduction of this new column, all the protocols are evaluated against this new characteristic. In general, these protocols will have a rating of 0 meaning that the protocol doesn't have this characteristic. Exceptionally, the evaluation of some protocols may produce non-zero ratings because our recommended classification of protocols against the characteristics is not exhaustive.

An exhaustive evaluation of a protocol against all the possible characteristics introduces overhead when compared to a selective evaluation against relevant constraints. Selecting a *relevant constraint* is based on two aspects:

1. The ratings of the other protocols and the potential new protocols against this constraint. The benefit of adding a new constraint neither supported by the new protocol nor by any of the already defined protocols is limited. However, if new protocols will support it, saving the rating of the current protocol against this constraint reduces the process time of the future reevaluation.

2. How often this constraint would appear as a constraint in the conflicts to be negotiated. If the characteristic is not supported by the protocol but is common in conflicts, having it in the protocol-constraint table improves the protocol selection of NSP.

Although the guidelines for building this set are subjective, the set itself is dynamic. So, if a characteristic of a protocol has been discarded in the initial integration process of the protocol, PS allows the reevaluation of the classification of the protocol and the proper adjustment of the protocol-constraint table. Although the rating of the protocols is done heuristically, this table constitutes a step forward in providing

automated guidance in selecting the negotiation protocols best suited for the conflict at hand.

Guidelines for Choosing a Rating:

For illustration purposes, we rate protocols approximately in this classification. Although, these numbers are not exact, they rate the protocols relatively in satisfying their constraints at a coarse level of granularity. Only when protocols exhibit close ratings for a specific constraint that rating is sensitive. The comparisons necessary to produce these exact ratings are beyond the scope of our research. Since these protocols have similar support for this constraint, the impact of choosing one protocol rather than the other is minimal.

We gather the characteristics or constraints to be satisfied by the selected protocol from the following sources:

- The problem and the environment the protocol is targeting.
- The quality and sophistication of the protocol's algorithm.
- The results guaranteed by the protocol.

This information can be gathered from the authors of the protocol and the various critiques the protocol received. With the spread of NSPs and their acceptance, releasing protocols along with their classification can lead to a defacto standard, making it faster and simpler to integrate those protocols to PS.

We provide the following high-level algorithm to build the Protocol-Constraint table.

Input:

1. Protocol Description
2. Protocol Implementation

Algorithm:

1. If the protocol is not already defined in the table
   (a) Add a new row to the table
   (b) Link the implementation of the protocol to its pointer in PCT
2. Build the characteristics list from

(a) the problem addressed by the protocol

(b) the targeted environment setting

(c) the algorithm

(d) the results of the protocol

3. For each characteristic in the table

   (a) If the characteristic is in the list

      i. assign it the rating in the list

      ii. remove it from the list

   (b) If the characteristic is not in the list, double check its rating

      i. If it is not supported, give it a rating of 0

      ii. If it is supported, give it the reviewed rating

4. For each remaining characteristic in the characteristics list

   (a) Add a new column to the table

   (b) Use the rating of this characteristic in the list for the new protocol

   (c) For each old protocol

      i. Evaluate the protocol against this column's characteristic

      ii. If the protocol supports the characteristic, assign it the proper rating

      iii. If the protocol does not support the characteristic, assign it a 0 rating

Output: Protocol-Constraint Table

5.3 Building the Protocol-Constraint Table

We will illustrate the building of the protocol-constraint table by following the Protocol-Constraint algorithm presented in the previous section to integrate the three classical protocols introduced in the first section of this chapter.

Initially, PCT is assumed empty as depicted in table 12:

Protocol Name	Function Name

Table 12: Initial Protocol-Constraint Table

As we mentioned earlier, with the integration of new protocols, the table grows both horizontally and vertically.

5.3.1 Contract Net Illustration

Using Contract Net as an example, we will go through the procedure of integrating a protocol into the Protocol-Constraint Table illustrated in table 11.

This protocol is the first one to be added to the table 12 and a new row is appended to the table to hold the characteristics of the Contract Net Protocol [step 1(a)]. The protocol name is entered in the first column: Contract Net. The function name is entered in the second column and it is effectively a pointer to the code of the function implementing the algorithm of this protocol [step 1(b)].

In step 2(a), we investigate the problem. As its objectives, the Contract Net Protocol supports both *task distribution* with a rating of *10* and *resource distribution* with *10*. In step 2(b), we check the targeted environment. The protocol is applied among loosely coupled entities hence it supports *a dynamic environment* with a rating of *10*.

In step 2(c), we analyze the quality and sophistication of the algorithm. Giving the agents the freedom in bidding, the protocol supports *load balancing*. A rating of *9* has been attributed because of the risk of zealous agents who could provoke a fake balanced load. The value 9 is chosen based on the **Guidelines for Choosing a Rating** presented in the previous section. Because roles are not mutually exclusive and any node may be both a manager and a contractor, the protocol supports *decentralized control* with a rating of *10*. The protocol is a two-way exchange and has the *multi stages* characteristic. A rate of *2* is assigned to reflect the number of stages it has. Contract Net does not detect conflicts. A rating of *0* is assigned to its *detect conflicts* constraint. The algorithm does not support *bargaining* since nodes cannot issue counterproposals and a rating *0* is assigned. No opportunity for *compromise* is available and this constraint gets a *0* rating. The protocol assumes its nodes to be both *benevolent agents* and *cooperative agents*. Both of these constraints get a rating

of *10*.

In step 2(d), we check the results guaranteed by the protocol. The multi-cast communication required by the protocol make it quite *communication intensive* with a rating of *9*. The algorithm does not provide *report generation* since the nodes do not communicate any explanations or reasoning for their stands. This constraint is assigned a *0* rating. The algorithm is not designed to be *optimal* so the rating is also *0*.

Protocol Name	Function Name	Dynamic Environment	Task Distribution	Resource Distribution	Load Balancing
Contract Net	CNet()	10	10	10	9

Protocol Name	Decentralized Control	Multi Stages	Detect Conflicts	Bargaining	Compromise
Contract Net	10	2	0	0	0

Protocol Name	Benevolent Agents	Cooperative Agents	Communication Intensive	Report Generation	Optimal
Contract Net	10	10	9	0	0

Table 13: Contract Net in the Protocol-Constraint Table

The resulting row in the Protocol-Constraint table is depicted in table 13. Note that we have split the table into three because it is too wide for a page.

Because this is the first protocol and no characteristics are already defined in the table, step 3 requires no action. In step 4, all the characteristics extracted are automatically incorporated into the table. For each new constraint, a new column is created [step 4(a)]. The rating of the Contract Net protocol against this constraint is entered in the column corresponding to the protocol and the constraint [step 4(b)]. Since this is the first protocol, there are no other protocols to evaluate, so step 4(c) requires no action.

5.3.2 Multistage Negotiation Protocol Illustration

The second illustration of the algorithm uses the Multistage Negotiation Protocol to the Protocol-Constraint table.

Step 1. The Multistage Negotiation Protocol is not defined in the table

Step 1. (a) A new row is added to the table.

Protocol Name	Function Name	Dynamic Environment	Task Distribution	Resource Distribution	Load Balancing
Contract Net	CNet()	10	10	10	9
Multistage	MSN()	10	10	10	9

Protocol Name	Decentralized Control	Multi Stages	Detect Conflicts	Bargaining	Compromise
Contract Net	10	2	0	0	0
Multistage	10	10	10	3	3

Protocol Name	Benevolent Agents	Cooperative Agents	Communication Intensive	Report Generation	Optimal
Contract Net	10	10	9	0	0
Multistage	10	10	9	0	4

Protocol Name	Global View	Detect Overconstraints	Subgoals Interactions	Conflicts of Interest
Contract Net	0	0	0	0
Multistage	0	10	10	0

Table 14: Multistage Negotiation Protocol in the Protocol-Constraint Table

Step 1. (b) Its implementation is linked to its function in the table.

Step 2. Because this protocol is a generalization of the Contract Net Protocol, it inherits the majority of its characteristics and their ratings. Each constraint is listed in the characteristics list of the Multistage Negotiation Protocol. However, to avoid repetition, we will only mention the constraints which either have a different rating than the one of the Contract Net or are newly added with this protocol.

In this generalization, the exchange is done iteratively so the constraint *multi stages* is attributed a rating of *10*. As opposed to the Contract Net, this protocol supports *detect conflicts* with a rating of *10*. Also *bargaining* rated at *3* and *compromise* rated at *3* have been introduced to the protocol. The two constraints *global view* and *conflicts of interest* have been added as not supported with the Multistage Negotiation Protocol. The other two constraints *detect overconstraints* and *subgoals interactions* are supported and hence were attributed a rating of *10* each. A rating of *4* is assigned to *optimal* since the algorithm does not guarantees any optimality and it is based on heuristics.

Step 3. The characteristics of the Contract Net not explicitly mentioned are attributed the same rating for this new protocol. The constraints with a different rating are also entered in the table.

Step 4. For characteristics remain in the list.

Step 4. (a) Four columns are added for *global view, conflicts of interest, detect overconstraints* and *subgoals interactions.*

Step 4. (b) The Multistage cells in the new columns are filled with the ratings of the list.

Step 4. (c) The Contract Net Protocol has been attributed *0* for these new four constraints.

5.3.3 Partial Global Planning Illustration

The final illustration uses the Partial Global Planning protocol to the table. Table 14 evolves into table 15.

Step 1. The Partial Global Planning is a new protocol

Step 1. (a) One more row is added to the table

Step 1. (b) The implementation of the protocol is linked to its pointer in the table.

Step 2. Only new characteristics or characteristics with different ratings will be presented here to avoid repetition. A stable environment is needed for nodes to converge on common plans and thus, *dynamic environment* is attributed a rating of *3.* The framework supports competing agents so *competing agents, benevolent agents, cooperative agents, misrepresentation, exerting authority* and *conflicts of interest* are given a rating of *5* each. As one of the different styles of cooperation in the framework, *centralized control* is given a rating of *10.*

Step 3. Not all the characteristics defined in the table are in the list.

Step 3. (a) Each characteristic in the table found in the list will be assigned the list's rating and then removed

Protocol Name	Function Name	Dynamic Environment	Task Distribution	Resource Distribution	Load Balancing
Contract Net	CNet()	10	10	10	9
Multistage	MSN()	10	10	10	9
Partial Global Planning	PGP()	3	10	10	9

Protocol Name	Decentralized Control	Multi Stages	Detect Conflicts	Bargaining	Compromise
Contract Net	10	2	0	0	0
Multistage	10	10	10	3	3
Partial Global Planning	10	10	10	3	3

Protocol Name	Benevolent Agents	Cooperative Agents	Communication Intensive	Report Generation	Optimal
Contract Net	10	10	9	0	0
Multistage	10	10	9	0	4
Partial Global Planning	5	5	9	0	4

Protocol Name	Global View	Detect Overconstraints	Subgoals Interactions	Conflicts of Interest
Contract Net	0	0	0	0
Multistage	0	10	10	0
Partial Global Planning	0	10	10	5

Protocol Name	Competing Agents	Misrepresentation	Exerting Authority	Centralized Control
Contract Net	0	0	0	0
Multistage	0	0	0	0
Partial Global Planning	5	5	5	10

Table 15: Partial Global Planning in the Protocol-Constraint Table

Step 3. (b) For example, *resource distribution* was not on the list of characteristics compiled for the Partial Global Planning protocol because this characteristic is not directly addressed in its description. After considering the possibility of *resource distribution* using the framework, it is assigned a rating of *10*.

Step 4. *Competing agents*, *misrepresentation*, *exerting authority* and *centralized control* are the new constraints introduced with this protocol.

Step 4. (a) A new column is created for each one.

Step 4. (b) The rating of the protocol against each of these characteristics is entered in the table.

Step 4. (c) The Contract Net then the Multistage Negotiation protocols are evaluated against each of these constraints and then assigned a rating. In these cases, a rating of *0* is assigned.

Chapter 6

Conclusion

In this thesis, we addressed the task of negotiation between software agents from a new angle. We developed a system called Negotiation Service Provider (NSP), that provides negotiation services to a community of agents. NSP makes use of a dynamic set of negotiation protocols and select an appropriate protocol for a given need. Agents interact with each other to resolve their conflicts without having the burden of conducting negotiation. Instead, the Negotiation Service Provider (NSP), we introduced in this thesis, will negotiate on their behalf.

6.1 Contributions Summary

The notion of providing negotiation as a service is innovative in the field of intelligent agents. Two previous research work similar to NSP in their scope are the system developed at Vrije Universiteit Amsterdam and University of Karlskrona/Ronneby [BCG+00, BCG+98] and Multi-Agent Negotiation Testbed (MAGNET) [CTMG98] developed at the University of Minnesota [CTMG98]. The former system will be referred as DESIRE in the following discussions. The following five characteristics provide a critical comparison of our NSP with the other two.

1. **NSP includes a dynamic suite of negotiation protocols.** DESIRE implements the monotonic concession protocol described by Rosenschein and Zlotkin to control the negotiation processes. For the negotiation, three different announcement methods are distinguished: the offer method, the request for bids method and the announce reward tables. MAGNET uses a "simple three step,

leveled commitment protocol". NSP includes a dynamic protocol suite which supports the integration of new protocols to this suite.

2. **NSP has a broad scope for negotiation.** DESIRE is specialized in dynamic load management whereas MAGNET negotiates based on temporal and precedence constraints and supports time-based contingencies. NSP is not focused to such a narrow level of specialization. Depending on the requirements, NSP uses its Protocol-Constraint Table (PCT) to choose the appropriate protocol to conduct the negotiation.

3. **NSP provides negotiation services to heterogeneous agents.** DESIRE can handle pre-specified types of agents such as utility agents, customer agents, production agents and resource consumer agents. MAGNET provides a generalized market architecture. Similar to MAGNET, NSP supports heterogeneous agents. Additionally, NSP can deal with multiple agent communication languages (ACL).

4. **NSP negotiates independent of the agent's domain of expertise.** DESIRE is specifically designed for the domain of electrical energy usage. On the contrary, the types of transactions supported in MAGNET vary from simple buying and selling goods and services to complex multi-agent contract negotiations. NSP puts no constraint on the application domains of its clients.

5. **NSP provides an infrastructure for negotiation.** DESIRE operates in the context of a closed system whereas MAGNET and NSP provide an infrastructure for negotiation in an open system environment. However, NSP is relatively more flexible in providing different styles of negotiation.

The methodology for dynamically adding a new protocol to the PS of a NSP we introduced is original. It spins around the Protocol-Constraint Table (PCT) used by negotiation agents (NA) to select the most advantageous negotiation protocol depending on the needs of the conflict. We achieved the "proof of concept" of our methodology by applying its algorithm to a test case of three protocols.

Analysis and classification of conflicts constitutes another contribution of this thesis. We analyzed conflicts and identified six different areas to describe a conflict: its context, its cause, the agent's own prioritized list of goals, constraints, margin of

maneuver, and information. The initial task of a negotiation agent (NA) is to gather as much details as possible in these six areas through dialogs with both its client and other agents involved in the conflict. Based on the constraints developed from the above knowledge, NA selects the negotiation protocol for its use.

6.2 Future Work and Open Issues

Because of the diversity in NSP, our research did not address all the different details and aspects of this system at the same level of depth. The following areas deserve further investigation:

- **Agent communication languages:**
  The field of ACL is not mature enough yet, languages like KQML and FIPA are still ambiguous and vague [NN99]. Further "accurate and complete translations" from between KQML and FIPA are not, in general, possible [YLP99]. To achieve agent inter-operability, further work is needed in this area. Questions such as "What happens if a command in a specific ACL is not available in the target language? Is the command disregarded? How is an approximation found?" need to be answered before the NSP's ACLS reaches its full potential.

- **Relationship between clients and NAs:**
  A more detailed research of the relationship between a client and its assigned NA is needed. Specifically, the dialogs between NA and the parties involved in the conflict require further investigation.

- **Report Generation & Error Logging:**
  To gain the confidence of its customers, the report generation & error logging functionalities in NSP must be developed. These functionalities allow external sources to evaluate and audit the performance of NSP.

Bibliography

[BCG+98] F.M.T. Brazier, F. Cornelissen, R. Gustavsson, O. Lindeberg C.M. Jonker, B. Polak, and J. Treur. Design and verification of a multi-agent system for one-to-many negotiation. In *Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS'98*, pages 49–56. IEEE Computer Society Press, 1998.

[BCG+00] F.M.T. Brazier, F. Cornelissen, R. Gustavsson, O. Lindeberg C.M. Jonker, B. Polak, and J. Treur. Multi-agent system performing one-to-many negotiation for load balancing of electricity use. *International Journal of Electronic Commerce*, 2000.

[CML98] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Reading in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, INC., 1998.

[CTMG98] J. Collins, M. Tsvetovat, B. Mobasher, and M. Gini. MAGNET: A multi-agent contracting system for plan execution. In *Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, pages 63–68. AAAI Press, 1998.

[CW94] Man Kit Chang and Carson C. Woo. A speech-act-based negotiation protocol: Design, implementation, and test use. *ACM Transactions on Information Systems*, 12(4):360–382, 1994.

[DL98] Edmund H. Durfee and Victor R. Lesser. Using partial global plans to coordinate distributed problem solvers. In Alan H. Bond and Les Gasser,

editors, *Reading in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, INC., 1998.

[DS83] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[ea97] J.M. Bradshaw et al. Kaos: Toward and industrial-strength open agent architecture. In J.M. Bradshaw, editor, *Software Agents*. AAAI Press/MIT Press, 1997.

[FFR93] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua server: A tool for collaborative ontology construction. Technical Report Tech. Report KSL-96-26, Stanfor Knowledge Systems Laboratory, Stanford, California, 1993.

[FLM97] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In J.M. Bradshaw, editor, *Software Agents*. AAAI Press/MIT Press, 1997.

[GC95] Leonard Greenhalgh and Deborah I. Chapman. Joint decision making the inseparability of relationships and negotiation. In Roderick M. Kramer and David M. Messick, editors, *Negotiation as a social process*, pages 166–185. Sage Publications International, 1995.

[KSE98] Sarit Kraus, Katia Sycara, and Amir Evenchik. Reaching agreement through argumentation: a logical model and implementation. *Artificial Intelligence*, 104:1–69, 1998.

[Lab96] Y. Labrou. *Semantics for an Agent Communication Language*. Doctoral dissertation, University of Maryland, Baltimore County, 1996.

[Len95] D.B. Lenat. Cyc: A large scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[LF97] Y. Labrou and Tim Finin. Semantics and conversations for an agent communication language. In M. Huhns and M. Singh, editors, *Reading in Agents*. Morgan Kaufmann, 1997.

[LF98] Y. Labrou and Tim Finin. Semantics for an agent communication language. In M. Woodridge, J.P. Muller, and M. Tambe, editors, *Agent*

Theories, Architectures and Languages IV. Springer-Verlag, 1998. Lecture Notes in Artificial Intelligence.

[NN99] Hyacinth S. Nwama and Divine T. Ndumu. A perspective on software agents research. *The Knowledge Engineering Review*, 14(2):1–18, 1999.

[Smi98] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. In Alan H. Bond and Les Gasser, editors, *Reading in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers INC., 1998.

[urla] Internetworking technology overview. Cisco Systems Inc. http://www.cisco.com/univercd/home/home.htm.

[urlb] Corba v2.2 section 13.7. Object Management Group. http://www.omg.org/corba/cichpter.html.

[urlc] Iiop. Webopedia. http://webopedia.internet.com/Computer_Science/Distributed_Computing/IIOP.html.

[urld] Creating a custom rmi socket factory. Javasoft. http://www.javasoft.com/products/jdk/1.2/docs/guide/rmi/rmisocketfactory.doc.html.

[urle] Organisation for Economic Co-operation and Development. http://www.oecd.org/dsti/sti/industry/indcomp/act/services.htm.

[urlf] Service provider and telecommunications serviceprovider.com. Sun Microsystems, Inc. http://www.sun.com/sp/serviceprovider/index.html.

[urlg] Service provider and telecommunications suntone. Sun Microsystems, Inc. http://www.sun.com/sp/suntone/index.html.

[urlh] Isp. AOL Webopaedia. http://aol.pcwebopedia.com/TERM/I/ISP.html.

[urli] How to .com case studies. Sun Microsystems, Inc. http://www.sun.com/dot-com/studies/boom.html.

[urlj] Application service provider. AOL Webopaedia. http://aol.pcwebopedia.com/TERM/A/Application_Service_Provider.html.

[urlk] Internetworking technology overview. Cisco Systems Inc. http://www.cisco.com/univercd/home/home.htm.

[url97] Fipa 97 specification part 2 agent communication language. FIPA, October 1997. http://www.fipa.org/.

[url98] Fipa 98 specification part 12 ontology service. FIPA, October 1998. http://www.fipa.org/.

[url99a] Analysis of 1998 u.s. employment data services-based economy means cheaper labor manufacturing. mining jobs shrink. Truth in Media Global Watch Bulletins. March 1999. http://www.truthinmedia.org/Bulletins99/tim99-3-3.html.

[url99b] Mindbridge.com launches application service provider initiative with intrasmart intranet in a box software. IntraSmart. July 1999. http://www.intrasmart.com/PRapplicationlaunch.htm.

[YLP99] Tim Finin Yannis Labrou and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent System*, 14(2):45–52, 1999.