Chapter-2

Communication Protocols & Principles

2.1 Introduction

An interesting characteristic of multi-agent systems is the principle that agents can function more effectively in groups. Agents are designed to autonomously collaborate with each other in order to satisfy both their internal goals and the shared external demands generated by virtue of their participation in agent societies. This type of collaboration depends on a sophisticated system of inter-agent communication. The language used by agents for this communication is the Agent Communication Language (ACL). The main objective of an ACL is to model a suitable framework that allows heterogeneous agents to interact and to communicate with meaningful statements that convey information about their environment or knowledge.

Over the last decade, two main ACLs have been proposed: the Knowledge Query and Manipulation Language (KQML) [1] and the Foundation for Intelligent Physical Agents’ Agent Communication Language (FIPA-ACL). FIPA-ACL is in turn based on the ARTIMIS Communication Language (ARCOL). The formal specifications and the semantics of these languages are based upon the philosophical foundation provided by Speech Act Theory. The purpose of this chapter is to introduce these languages/protocols and their philosophical foundations.

2.2 KQML

KQML arose from work sponsored by the American Government’s Defense Advanced Research Projects Agency (DARPA). It is the result of research done by the Knowledge Sharing Effort (KSE), an initiative that aims at developing a foundation for
software systems interaction and interoperability. Three working groups compose this consortium: the Interlingua group, the shared and reusable knowledge base group, and the external interface group. The first group designed the Knowledge Interchange Format (KIF) as a common language for describing message content. This format is an extension of first-order logic. The second group worked on the content of sharable knowledge bases. This group examined the problem of sharing the content of formally represented knowledge. The approach focused on common ontologies. Every knowledge-based system relies on some conceptualization of the world (objects, qualities, distinctions and relationships that matter when performing some task) that is embodied in concepts, distinctions, etc. using a formal representation. The group worked on the construction of ontologies for various domains. Each ontology, written in KIF, defines a set of classes, functions, and objects for some domain of discourse, and includes an axiomatization to constrain the interpretation. The third group produced the KQML language and looked at interactions of system components.

The language’s primitives are called *performatives*. As the term suggests, the concept is related to *speech act theory* [2]. Performatives define the permissible actions (operations) that agents may attempt when communicating with each other. KQML consists of a set of communication primitives aiming to support interaction between agents. In this language, an agent’s mental attitudes (belief, intention, and desire) are expressed in the label of a message that represents a communicative act. A KQML message is conceptually divided into three levels [3]: (1) the communicative level which specifies the sender and receiver agents; (2) the message level which mainly specifies the type of performatives (affirmation, question, etc.), the language (KIF, Prolog, etc.) and the used ontology; (3) the content level, which specifies the message content. An example of KQML message is the following:
The goal of the tell KQML performative is to convey to some receiving agent that the sending agent believes that a particular proposition (contained in the content field) is true. The example indicates that agent \( X \) answers a message of agent \( Y \) about the price of a software. It uses the Prolog language to describe the content and a particular ontology (Software) which indicates the significance of "MathType" and the currency associated with the value "150".

Initially, KQML was proposed without a defined semantics. This criticism led researchers to define a new language: the FIPA-ACL. The early version of KQML presented some confusions and ambiguities in the usage of the performatives. Later on, its authors gave it a semantics and limited the use of some performatives in order to avoid some of these problems. The new semantics is defined in terms of: 1) preconditions on the mental states of the sender and the receiver before the communication of the message, 2) postconditions that should hold after the message is sent and 3) completion conditions that indicate when the perlocutionary effect has been fulfilled. However, this semantics provides no semantic model for mental attitudes.

### 2.3 ARCOL

The ARTIMIS agent technology developed by France Telecom is a generic framework for instantiating communicating agents. This technology is based on the proposal of Sadek [4]. In ARTIMIS, an agent can cooperatively interact with humans as
well as with other agents. Agents’ communicative acts are modeled as rational actions. Agents can reason about knowledge and actions pertaining to their communicative acts. ARCOL (ARTIMIS Communication Language) is the ACL used in ARTIMIS. An ARCOL expression relies on a semantic language SL for the definition of its semantics. SL, in turn, uses the language SCL (Semantic Content Language) to describe the semantics content of a communicative act. ARCOL includes the following set of primitives:

*Inform*: An agent uses the assertive act Inform to convey a message to another agent provided that it believes the content of this message.

*Request*: The directive act request enables an agent to demand an action from another agent provided that it has the capabilities to perform that action.

*Confirmation*: When the sender believes that the receiver is uncertain about the information being transmitted, this communicative act can be used to confirm it.

*Inform referent*: This communicative act enables an agent to inform another agent of the value of a referent with a given description.

The most important characteristic of the ARCOL language is its formal semantics as a reliable support for interoperability. However, ARCOL’s fixed context with the sender agent required to be sincere is an impediment to heterogeneity.

### 2.4 FIPA-ACL

FIPA-ACL arose from attempts to develop an industry standard for agent communication. Its formal model and semantic language draw from the ARCOL Language. Conceptually, FIPA-ACL distinguishes two levels in communication messages. At the inner level, the content of messages can be expressed in any logical
language. The outer level describes the locutions that agents can use in their communication. The content of messages is wrapped in these locutions. FIPA-ACL specifies 22 locutions [5]. Here we mention some of them:

*Accept Proposal*: The action of accepting a previously submitted proposal to perform an action.

*Agree*: The action of agreeing to perform some action, possibly in the future.

*Call for Proposal*: The action of calling for proposals to perform a given action.

*Confirm*: The sender informs the receiver that a given proposition is true, where the receiver is known to be uncertain about the proposition.

*Inform*: The sender informs the receiver that a given proposition is true.

*Not Understood*: The sender informs the receiver that it received a message that it does not recognize or it is unable to process the content of this message.

*Propose*: The action of submitting a proposal to perform a certain action, given certain preconditions.

*Query If*: The action of asking another agent whether or not a given proposition is true.

*Request*: The sender requests the receiver to perform some action or a communicative act.

FIPA-ACL is an agent communication language whose development involved several parties in industry and academia. It lays out the practical components of agent communication and cooperation and a well-defined formal semantics. However, some practical applications pointed out several limitations of the FIPA standard [6]. For
example, this standard provides no support for real-time and performance requirements of telecommunication applications. In addition, FIPA-ACL semantics rests only on the belief states of communicative agents. In this context, the sender does not guarantee the actual accomplishment of the expected outcome at the destination because the semantics offers no mechanism on how to infer the mental state of the receiving end.

### 2.5 A Taxonomy of ACL Semantics

When considering formal languages, different semantics can be defined viewing them as a mathematical logic. A semantics is a relationship between the language and a space $M$ of mathematical structures, called *models*. A statement $S$ in the language specifies a subset $M(S)$ of $M$. Such a statement is said to be true in a particular model $M_0$ if $M_0 \in M(S)$. A statement is said to be logically valid if it is true in every model, i.e., if $M(S) = M$.

Another type of semantics is derived from linguistics. As expressed by the linguist Morris [7], the syntax of a language is *the formal relation of signs to one another* and the semantics of the language defines *the relations of signs to the objects to which the signs are applicable*. Thus, it makes sense to speak of the truth of a sign, since this indicates that the sign has a relationship to external objects in the world [8].

Because ACLs are formal languages, their semantics can be defined as in mathematical logic. However, because they are also intended as communication mechanisms, a linguistics-based semantics can be defined. In this work, formal semantics defined from mathematical logic is being presented. Five formal semantics are: *axiomatic semantics*, *operational semantics*, *denotational semantics*, *game-theoretic semantics*, and *tableau semantics*. 
Axiomatic semantics is defined by a set of assertions about properties of a system and how they are affected by program execution. For ACLs, this semantics defines each locution in terms of the preconditions which must be fulfilled before the locution can be uttered, and in terms of the post-conditions which must become true after the production of the utterance. There are two types of axiomatic semantic i.e. public and private axiomatic semantics. In public axiomatic semantics, the pre-conditions and post-conditions describe states or conditions of the dialogue which can be observed by all participants. In private axiomatic semantics, pre and post-conditions describe states or conditions which are internal to one or more of the participants and thus are not directly observable by the others. For example, the semantics defined for FIPA-ACL and KQML are private axiomatic semantics. For example, the Inform FIPA-ACL act, in which one agent tells another some proposition, may only be uttered if the first agent believes the proposition to be true. This is termed a sincerity condition. On the other hand, the semantics provided for argument-based ACLs is a public axiomatic semantics [9]. For example, according to this semantics, an agent which asserts a proposition is supposed to have an argument in favor of it. Thus, if this proposition is attacked by another agent, agent must defend it.

Operational semantics is defined by a set of rules specifying how the state of an abstract machine changes while executing a set of instructions. Each rule specifies certain preconditions on the contents of states and their new contents after the application of the rule. In the context of ACLs, operational semantics considers the locutions as instructions which operate successively on the states of some abstract machine. This semantics defines the locutions in terms of the transitions they apply on the states of this machine. van Eijk and his colleagues (2000) studied operational semantics for ACLs on the basis of an agent communication programming language which is a formal framework that identifies basic aspects of agent communication. The formal semantics of this language is given by means of transition rules that describe its operational behavior. Moreover, the
operational semantics closely follows the syntactic structure of the language, and hence gives rise to an abstract machine to interpret the language.

Denotational semantics is a technique for describing the meaning of programs in terms of mathematical functions. Programs are translated into functions whose properties can be proved. In denotational semantics, each element of the language syntax attributes a relationship to an abstract mathematical entity, its denotation. The possible world semantics related to modal languages is an example of such semantics [10, 11]. For example, the semantics of the necessarily and possibly modal connectives are given by introducing an accessibility relation into models for the language. This relation defines what worlds are considered accessible from every other world. A formula is necessarily true if it is true in every world accessible from the current world, and it is possibly true if it is true in at least one world accessible from the current world. An example is given in [12]. In this semantics, argumentation systems are connected to qualitative probabilistic networks. Propositions correspond to nodes in these networks and arguments between propositions correspond to the associated nodes. In order to use the denotational semantics approach, we must be able to derive the semantic meaning of a statement expressed in the language from the semantic meaning of its elements, this property is called compositionality.

In game-theoretic semantics, each well formed formula in a language is associated with a formal game between two players: a protagonist and an antagonist. A statement is considered to be true when and only when a winning strategy exists for the protagonist in the associated game. In [8], a game-theoretic semantics for an inquiry dialogue protocol is proposed. In this semantics, a winning strategy for a player is a set of rules enabling the player to move in such a way that executing these moves guarantees that the player can win the game, no matter which moves are made by the opposing player.
Tableau semantics is based on the use of assertions and proof rules. The proof rules are inference rules aiming to prove the truth or falsity of the assertions. Unlike traditional proof systems which are bottom-up approaches, tableau semantics uses a top-down or goal-oriented approach. Proof rules are used in order to prove a certain formula by inferring when a state in a Kripke structure satisfies such a formula. According to this semantics, we start from a goal, and we apply a proof rule and determine the sub-goals to be proven. The proof rules are designed so that the goal is true if all the sub-goals are true. In ACLs, this semantics can be used to give the meaning of communicative acts by considering them as goals, and then determining the sub-goals by applying a set of proof-rules.

2.6 Conversation Protocols

When conversing, agents do not exchange isolated messages, but a sequence of interdependent messages. To take into account this aspect, FIPA proposes to use conversation protocols (also called conversation policies). Conversation protocols are general constraints on the sequences of semantically coherent messages leading to a goal [13]. The coherence of messages is ensured by these constraints. These protocols are specified as static structures which define in a deterministic way the order in which communicative acts are connected. Like protocols used in distributed systems, these structures are generally modeled using finite state machines [14, 15] Petri nets (traditional or extended) [16] or UML sequence diagrams. The idea of protocols is to facilitate the task of computing the possible answers to a given message. Request Interaction Protocol, Contract Net Interaction Protocol, and English Auction Interaction Protocol (FIPA, 1997, 1999, 2001a) are examples of such protocols.

As outlined by Greaves and his colleagues (2000) and Vongkacem and Chaib-draa (2000), conversation protocols must address two fundamental issues:
**Flexibility** : The aim of conversation protocols is basically to constrain the conversational behavior of the participants while taking into account the fact that agents are autonomous. These protocols must find equilibrium between the normative aspect ensured by the constraints and the flexibility expected in multi-agent communications.

**Specification** : Conversation protocols must be specified while taking into account the computational complexity of reasoning about them. For example, this specification must avoid the state-explosion problem when analyzing a sequence of utterances in order to decide which locution to utter next. In addition, these protocols should be designed in such a way that a formal verification is possible. The verification of some properties in these protocols, for example, deadlock, termination, correctness, etc. is extremely important in open environments.

2.7 From the Philosophy of Language to Agent Communication

2.7.1 From Speech Act Theory to Conversations

The specifications of KQML, ARCOL, and FIPA-ACL are based on a philosophical theory called *speech act theory*. This theory is due originally to a philosopher of language [2], and extended by [17] and Searle and [18]. It considers human utterances as actions, in that they may change the state of the world. Speech is not just used to designate something, it actually does something. This explains the use of the word "act" in the description of ARCOL and FIPA-ACL locutions. According to Searle, to understand language, one must understand the speaker’s intention. Since language is intentional behavior, it should be treated like a form of action. Thus, Searle refers to utterances as speech acts. The speech act is the basic unit of language used to express meaning, an utterance that expresses an intention. In general, speech acts are acts of communication. To communicate is to express a certain attitude, and the type of speech act being performed corresponds to the type of attitude being expressed. For example, a
statement expresses a belief, a request expresses a desire, and an apology expresses regret. As a communicative act, a speech act succeeds if the audience identifies, in accordance with the speaker's intention, the attitude being expressed.

Speech act theory identifies three distinct levels of action beyond the act of utterance itself. It distinguishes the act of saying something (the "locutionary" act), what one does in saying it (the "illocutionary" act), and what one does by saying it (the "perlocutionary" act). Speech acts, being perlocutionary as well as illocutionary, generally have some ulterior purpose, but they are distinguished primarily by their illocutionary type, such as asserting, requesting, promising and apologizing, which in turn are distinguished by the type of attitude expressed. The perlocutionary act is a matter of trying to get the hearer to form some correlative attitude and in some cases to act in a certain way. For example, a statement expresses a belief and normally has the further purpose of getting the addressee form the same belief. A request expresses a desire for the addressee to do a certain thing and normally aims for the addressee to intend to and, indeed, actually do that thing. A promise expresses the speaker's firm intention to do something, together with the belief that by his utterance he is obligated to do it, and normally aims further for the addressee to expect, and to feel entitled to expect, the speaker to do it.

As outlined in [18], speech act theory tends to study isolated illocutionary acts performed by using sentences in single context of utterance. However, it is clear that speech acts are seldom performed alone. Speakers perform their illocutionary acts within entire conversations in order to achieve common goals such as discussing news, coordinating their joint actions or negotiating. For this reason, Vanderveken proposed a theory of discourse enriching Speech Act Theory. The purpose of this theory is to analyze the structure of conversations whose type is provided with an internal discursive purpose and to provide a taxonomy of these conversations. This taxonomy is based on the fact that
there are only four possible discursive goals that speakers can attempt to achieve by way of conversing: the descriptive, deliberative, declarative, and expressive goals. These goals correspond to one of the four possible directions of fit between words and things. Using these directions, the four conversation types can be described as follow:

1. **Conversations with the words-to-things direction of fit have the descriptive goal** : They serve to describe what is happening in the world. Such are descriptions, debates on a question, persuasions, arguments, explications, interrogations, etc.

2. **Conversations with the things-to-words direction of fit have the deliberative goal** : They serve to deliberate on which future actions speakers and hearers should commit themselves to in the world. Such are deliberations, negotiations, bargaining sessions, a compromise or the signing of a contract, auctions, etc.

3. **Conversations with the double direction of fit have the declarative goal** : They serve to transform the world by way of doing what one says. Such are official declarations like declarations of war or of independence, nominations, appointments, etc.

4. **Conversations with the empty direction of fit have the expressive goal** : They serve to express common attitudes of their speakers. Such are the exchanges of greetings, welcomes, congratulations, etc.

**2.7.2 Walton and Krabbe’s Classification**

Another taxonomy of dialogues was proposed by two philosophers of argumentation, [19]. In their book: *Commitment in Dialogue*, *Basic Concepts of Interpersonal Reasoning*, Walton and Krabbe distinguish six main types of dialogues:

1. **Persuasion**, which is centered around conflicting points of view.
2. **Negotiation**, in which participants aim to achieve a settlement that is particularly advantageous for individual parties.

3. **Inquiry**, in which the aim is to collectively discover more information, as well as to destroy incorrect information.

4. **Deliberation**, which is driven by the need to take a collective decision.

5. **Information-seeking**, in which one party asks for information known by another.

6. **Eristic**, in which two parties combat each other in a quarrel.

While Vanderveken’s classification is based on directions of fit between words and things, Walton and Krabbe’s classification is based upon two factors: the initial situation and the goal of the dialogue. Table 2.1 illustrates these factors.

<table>
<thead>
<tr>
<th><strong>Dialogue Type</strong></th>
<th><strong>Initial Situation</strong></th>
<th><strong>Dialogue Goal</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Persuasion</td>
<td>Conflicting point of view</td>
<td>Resolution of Conflict</td>
</tr>
<tr>
<td>Negotiation</td>
<td>Conflict of interest</td>
<td>Making a deal</td>
</tr>
<tr>
<td>Inquiry</td>
<td>General ignorance</td>
<td>Growth of knowledge</td>
</tr>
<tr>
<td>Deliberation</td>
<td>Need for action</td>
<td>Reach a decision</td>
</tr>
<tr>
<td>Information-seeking</td>
<td>Personal ignorance</td>
<td>Spreading knowledge</td>
</tr>
<tr>
<td>Eristic</td>
<td>Antagonism</td>
<td>Accommodation in relationship</td>
</tr>
</tbody>
</table>

These six types may be refined into subtypes, simply by specifying more elaborate conditions on the dialogues (e.g. the type of conflict or the degree of rigidity of the rules). For example, a dispute is a subtype of persuasion, where each participant tries to defend
its point of view. In addition, this taxonomy is based on an argumentation vision and it coincides with the dialectical systems proposed in [20].

Walton and Krabbe introduced the notion of dialectical shift to capture the change in the context of dialogue during a conversation from one type of dialogue to another. Indeed, dialogues are usually not of a single type from their beginning to their end. For instance, it is common to start an inquiry dialogue, to realize during the dialogue that there is a controversial issue at stake, to enter in a dispute sub-dialogue, and to eventually resume the inquiry dialogue when the issue has been resolved. This notion allows us to construct complex dialogues by combining different types.

Vanderveken’s classification can be regarded as more general than Walton and Krabbe’s one in the sense that the dialogue types discussed by Walton and Krabbe are subtypes of the four types proposed by Vanderveken. Persuasion, Inquiry, and Information-seeking are conversations with a descriptive goal, negotiation and deliberation are conversations with a deliberative goal, and Eristic is a conversation with an expressive goal.

### 2.8 Contract Net Protocol

Contract Net Protocol is a high level protocol which is concerned with the interpretation of the communication rather than the transmission of bit streams. It supports communication among agents in distributed MAS. High level protocols provide structure that help system designers decide what the agents should say while communicating rather than how they should say it. CNP being high level protocol facilitates distributed control of cooperative task execution with efficient inter node communication and also allows participation in fully automated competitive negotiations. Each agent in the system takes on one of the role in the execution of task, which are manager role or contractor role. A manager is responsible for monitoring the execution of a task and processing the results of its execution. On the other hand, a contractor is
responsible for the actual execution of the task. The manager broadcast a message of task to be executed to all of contractors. Each message in the CNP has a set of slots for task-specific information such as task eligibility specification, task abstraction, bid specification, expiration time which are described as below:

- *Task eligibility specification* is a list of criteria that the contractor must meet to be eligible to submit a bid. The purpose of this slot is reducing message traffic by pruning contractors whose bids would clearly be not accepted.
- *Task abstraction* is a brief description of the task to be executed. It enables the contractor to rank the task relative to other announced tasks. The task abstraction is used rather than a full description of a task in order to reduce the length of the message.
- *Bid Specification* is a description of the expected form of a bid. It prevents the contractor from submitting the bid with irrelevant description of the task. Therefore, it will simplify the task of the manager in evaluating bids.
- *Expiration time* is a deadline for receiving bids

Contract Net Protocol specifies the interaction between agents for fully automated competitive negotiation through the use of contracts. In essence, Contract Net allows tasks to be distributed among a group of agents. In 1980 it was first applied to simulated distributed acoustic sensor network. One of the most promising use of Contract Net Protocol is to create an electronic marketplace for buying and selling goods. An important idea to note is that each agent is self-interested, meaning that the final solution may be the best for the agents involved, but not for the group as a whole.

Now, in CNP, various software agents make a network wherein each agent node can act as Manager (Initiator) or Contractor (Participant) performing different tasks at different times. Contract Net creates a means for contracting as well as subcontracting tasks (or jobs). In fact a manager is responsible for monitoring the execution of a task and
processing the results of its execution. On the other hand, a contractor is responsible for the actual execution of the task.

Contract Net Protocol performs at its best in all or either of the following three scenarios:

- The task can be decomposed into subtasks
- The subtasks are mutually independent
- The task is specific and hierarchical in nature

Contract Net Protocol can be advantageous when compared with other coordination strategies because of following reasons:

1. Tasks are assigned (contracts awarded) dynamically, resulting in the better deals for the agents involved.
2. Agents can enter and leave the system at will.
3. The tasks will be naturally balanced among all the agents since agents that already have contract(s) don’t have to bid on new ones. If an agent is already using all its resources, it will be unable to bid on new contracts until the current ones are completed.
4. A reliable strategy for distributed applications with agents that can recover from failures.

### 2.8.1 Need of CNP

Every agent in multi agent system is questionable to manager agent responsible for the completion of the task. For this, it formulates policies and procedures that are further executed by the rest of agents. It is main or imperial body of any system. The main feature of this system is that the manager agent has full control on its participants. MAS have caught more and more attention in recent years. These systems have been applied in variety of domains such as manufacturing, electronic commerce, traffic control etc. In MAS an agent is an entity that is situated in some environment and is capable of
acting autonomously in this environment in order to meet its design objectives. In MAS one agent cannot solve the complex problem solely since it has no sufficient competence, resources or information. For this they have to negotiate with each other that require a protocol [6]. The protocol is a set of rules agreed among the members of the system for their interactions and communication. The Contract Net Protocol (CNP) provided by Smith and Davis is often used for communication among the nodes in a network system during problem solving.

Next section discusses the architecture of CNP in detail.

2.8.2 Architecture/working of CNP

The working of Contract Net Interaction Protocol is composed of a sequence of four main steps. The agents must go through the following loop of steps to negotiate each contract.

1. The Initiator sends out a Call for Proposals (CFP).
2. Each Participant reviews CFP’s and bids on the feasible ones accordingly.
3. The Initiator chooses the best bid and awards the Contract to the respective Participant.
4. The Initiator rejects the other bids.

In a community, interactions between the master Web-service and the slave Web-services are framed in accordance with the contract-net protocol. The main purpose of these interactions is to identify the slave Web service that will take part in a composite Web service. The master-slave interactions of CNP are explained in the figure 2.4.
These steps will be explained further by elaborating the stages of the working of CNP. The high level view of CNP is shown in the fig.2.5 given below.

Fig.2.1: Master-Slave interaction of CNP

Fig.2.2: High Level view of CNP
By looking at above figure we can see that the manager agent performs its tasks in the form of stages. So the CNP typically has the five stages/phases. These stages are explained further.

1) Task Announcements

A node that generates a task normally initiates contract negotiation by advertising existence of that task to the other nodes with a task announcement message. It then acts as the manager of the task. A task announcement can be addressed to all nodes in the net (general broadcast), to a subset of nodes (limited broadcast), or to a single node (point-to-point). The latter two modes of addressing, which we call focused addressing, reduce message processing overhead by allowing non addressed nodes to ignore task announcements after examining only the addressee slot. The saving is small, but is useful because it allows a node’s communication processor alone to decide whether the rest of the message should be examined and further processed. It is also useful for reducing message traffic when the nodes of the problem solver are not interconnected with broadcast communication channels.

A task announcement has four main slots that are as follow:

(a) Eligibility Specification: It is a list of criteria that a node must meet to be eligible to submit a bid. This slot reduces message traffic by pruning nodes whose bids would be clearly unacceptable. In a sense, it is an extension to the addressee slot. Focused addressing can be used to restrict the possible respondents only when the manager knows the names of appropriate nodes. The eligibility specification slot is used to further restrict the possible respondents when the manager is not certain of the names of appropriate nodes, but can write a description of such nodes.
(b) **Task Abstraction:** It is a brief description of the task to be executed. It enables a node to rank the task relative to other announced tasks. An abstraction is used rather than a complete description in order to reduce the length of the message.

(c) **Bid Specification:** It is a description of the expected form of a bid. It enables the manager to specify the kind of information that it considers important about a node, interested in executing task. This provides a common basis for comparison of bids and enables a node to include in a bid only the information about its capabilities that is relevant to the task, rather than a complete description. This both simplifies the task of the manager in evaluating bids and further reduces message traffic.

(d) **Expiration Time:** It is a deadline for receiving bids. We assume global synchronization among the nodes. However, time is not critical in the negotiation process.

2) **Processing of Announced Tasks**

For each type of task, a node maintains a rank-ordered list of announcements that have been received and have not yet expired. Each node checks the eligibility specifications of all task announcements that it receives. This involves ensuring that the conditions expressed in the specification are met by the node. If it is eligible to bid on a task, then the node ranks that task relative to others under consideration.

3) **Submitting Bids**

This announcement-ranking activity proceeds concurrently with task processing in a node until the task processor completes processing of its current task and becomes available for processing another task. At this point, the contract processor is enabled to submit bids on announced tasks. It checks its list of task announcements and selects a task on which to submit a bid. If there is only one type of task, the procedure is
straightforward. If, on the other hand, there are a number of task types available, the node must select one of them.

An idle node can submit a bid on the most attractive task when either of the following events occur: 1) the node receives a new task announcement or 2) the expiration time is reached for any task announcement that the node has received. At each opportunity, the node makes a (task-specific) decision whether to submit a bid or wait for further task announcements. (In the signal task, a potential contractor waits for further announcements in an attempt to find the closest manager.)

The node abstraction slot of a bid is filled with a brief specification of the capabilities of the node that are relevant to the announced task. It is written in the form indicated by the bid specification of the corresponding task announcement.

4) Bid Processing/Contract Awarding

Contracts are queued locally by the manager that generated them until they can be awarded. The manager also maintains a rank-ordered list of bids that have been received for the task. When a bid is received, the manager ranks the bid relative to others under consideration. If, as a result, any of the bids are determined to be satisfactory, then the contract is awarded immediately to the associated bidder. (The definition of satisfactory is task-specific.) Otherwise, the manager waits for further bids.

If the expiration time is reached and the contract has not yet been awarded, several actions are possible. The appropriate action is task-specific, but the possibilities include: awarding the contract to the most acceptable bidder(s); transmitting another task announcement (if no bids have been received); or waiting for a time interval before transmitting another task announcement (if no acceptable bids have been received). This
is in contrast to the traditional view of task allocation where the most appropriate node available at the time would be selected.

Successful bidders are informed that they are now contractors for a task through an announced award message. The task specification slot contains a specification of the data needed to begin execution of the task, together with any additional information requested by the bidder.

5) **Contract Execution**

Contract Execution contains three slots stated ahead.

(a) **Contract Processing:** Once a contract has been awarded to a node, the information message is used for general communication between manager and contractor during the processing of a contract. The report is used by a contractor to inform the manager (and other report recipients, if any) that a task has been partially executed (an interim report) or completed (a final report).

(b) **Result Description:** This slot contains the results of the execution. Final reports are the normal method of result communication. Interim reports, however, are useful when generator-style control is desired. A contractor can be set to work on a task and instructed to issue interim reports whenever the next result is ready. It then suspends the task until it is instructed by the manager to continue (with an information message) and produce another result.

(c) **Termination:** The manager can also terminate contracts with a termination message. The contractor receiving such a message terminates execution of the contract indicated in the message and all of its outstanding subcontracts.
2.8.3 FIPA: CNP

The Foundation for Intelligent Physical Agents (FIPA) has specification for Contract Net Interaction Protocol as well as for communication between agents. The FIPA representation of Contract Net is shown below in Figure 2.6.

Figure 2.3 - FIPA Contract Net Interaction Protocol
The Initiator solicits m proposals from other agents by issuing a call for proposals (CFP) act, which specifies the task, as well any conditions the Initiator is placing upon the execution of the task. Agents (Participants) receiving the call for proposals are viewed as potential contractors and are able to generate n responses. Of these, j are proposals to perform the task, specified as propose acts. The Participant’s proposal includes the preconditions that the Participant is setting out for the task, which may be the price, time when the task will be done, etc. Alternatively, the i=n-j participants may refuse to propose. Once the deadline passes, the Initiator evaluates the received j proposals and selects agents to perform the task; one, several or no agents may be chosen. The l agents of the selected proposal(s) will be sent an accept proposal act and the remaining k agents will receive a reject proposal. The proposals are binding on the participant, so that once the initiator accepts the proposal; the participant acquires a commitment to perform the task. Once the participant has completed the task, it sends a completion message to the initiator in the form of an inform-done or a more explanatory version in the form of an inform-result. However, if the participant fails to complete the task, a failure message is sent. Note that this interaction protocol requires the initiator to know when it has received all replies. In the case that a participant fails to reply with either a propose or a refuse act, the initiator may potentially be left waiting indefinitely. To guard against this, the CFP includes a deadline by which replies should be received by the Initiator. Proposals received after the deadline, are automatically rejected with the given reason that the proposal was late. The deadline is specified by the reply-by parameter in the ACL message.

In order to simplify communications each message will contain a certain set of parameters. These parameters will include the type of message, the Initiator ID, the Participant ID, and the Conversation ID as well as the details of the message. The structure of the message would be (<Conversation ID>;<Message Type>;<Sender>;<Receiver>;<Content 1>;<Content 2>;...), where the content items are the details of the message. In the case of a communications error, where part of a
message is lost or the message doesn’t make sense, a message can be sent back to the sender easily indicating which message was not understood.

The message could also be specified in KQML (Knowledge Query and Manipulation Language). KQML messages contain the parameters from, to, reply-with, ontology, language and content. An example KQML message would be of the form:

\[(\text{monitor: from customer: to supplier: reply-with update-111: ontology standard-units-and-dimensions: language KIF: content } (= (q.magnitude (diameter shaft-a) inches) ?x))\]

By using FIPA standards for the design of a local-area Contract Net we can help to ensure that our network of agents is compatible with most other Contract Nets. As a result, the integration process will be much easier, allowing for the construction of broad-spectrum Contract Net systems spanning the globe.

FIPA Standardized CNP is the most widely used communication protocol for MAS communication. The following section highlights some of the applications of CNP.

2.8.4 Applications of CNP

Following are some application areas in which contract net protocol is extensively used:

i) E-Commerce: Agents use FIPA Iterated Contract Net Interaction Protocol that is revised with xml tag keeping original formal model causing to represent knowledge suitably in Internet environment. There are several advantages to use xml.

- It is simple to represent knowledge.
- It is convenient to deliver the message through http.
- It needs not to develop ACL parser as using built-in parser.
- It is easy to parse ACL encoded xml.
The selling agents reside in Internet shopping mall and human user use Internet. Starting to negotiate the buying agents sends CFP to selling agents for negotiation. And then the selling agents ‘propose’ 100-dollar for a bag of rice containing contract number in <in-reply-to> tag. The buying agent can ‘accept-proposal’ or ‘reject-proposal’ and in the case of ‘reject proposal’ it fills out the reason, then the selling agent proposes new CFP, after several giving and taking these message the buying agent finally ‘inform’ the price and terminate this procedure.

ii) **Task-Sharing and Contract Negotiation:** The contract net protocol facilitates cooperation of multiple processors in the solution of a problem. Dynamic matching of tasks and Knowledge sources is effected by negotiation. A contract is an explicit agreement between a processor that generates a task (the manager) and a processor willing to execute the task (the contractor). A contract is normally established by a process of local mutual selection based on a two-way transfer of information. In brief, available contractors evaluate task announcements made by several managers until they find one of interest. They submit a bid for that task. The manager then evaluates the bids received from potential contractors and selects the one it determines to be most appropriate. Both parties to the agreement have evaluated the information supplied by the other and a mutual selection has been made. Control is distributed because processing and communication are not focused at particular processors, but rather every processor is capable of accepting and assigning tasks.

Contract net messages contain slots for information that aids negotiation. A task announcement contains three such slots. The eligibility specification is a list of criteria that a processor must meet to be eligible to submit a bid. It enables a processor receiving the message to decide whether or not it is able to execute the task. This specification reduces message traffic by pruning processors whose bids would be clearly unacceptable. The task abstraction is a brief description of the task to be
executed. It enables a processor to rank the announced task relative to other announced tasks. An abstraction is used rather than a complete description in order to reduce the length of the message. The bid specification is a description of the expected form of a bid. It enables a processor to include in a bid only the information about its capabilities that is relevant to the task rather than a complete description (called a node abstraction). This simplifies the task of the manager in evaluating bids and further reduces message traffic.

iii) **Distributed Problem Solving:** In order to understand the motivations behind the Contract Net Protocol and the negotiation mechanism, it is worthwhile to first describe the problems these methods address. These methods fall under the umbrella of distributed problem solving the core idea behind multi-agent system systems. This is the approach of dividing a problem in such a way that different processors or different agents can work on smaller subsections of it. The most obvious reason for taking this approach is that using multiple processors can (depending on the problem and the way in which it is partitioned among the problem solvers) allow a solution to be found more quickly. Having the agents cooperate towards a common goal can also have benefits in terms of reliability. A key aspect here is that the agents do in fact cooperate, and not simply work in relative isolation. However, in contract net approach the agents are loosely coupled. Another aspect of distributed problem solving is that they are decentralized, meaning that control and data are dispersed logically among the problem solvers (which may be geographically dispersed as well). Since each agent has to work on a sub-section of the problem and complete its task as quickly as possible, it is desirable for that task to be fairly self-contained. This would allow the agent to spend most of its time focused on executing the task rather than retrieving data from another location or in communication with another agent.

iv) **Cellular Manufacturing System:** The Contract Net Protocol as being a dynamic scheduler is widely used in cellular manufacturing systems. In this approach, when an
operation of a job at a cell is finished, the cell’s control unit will make the decision regarding which cell the job should visit next. To do that, the cell’s control unit broadcasts the task announcements to the other cell control units. The cell control unit who received a task announcement checks if the required operation is within its capability and submits its estimation on the earliest finishing time (EFT) or shortest processing time (SPT). There is no job agent in this case. Each job’s route is determined through the negotiation between the cells. Shaw’s experimental results indicated that the bidding scheme with EFT (earliest finishing time) outperformed the bidding scheme with SPT (shortest processing time).

2.8.5 Shortcomings of CNP

Unfortunately, Contract Net Protocol also has a number of shortcomings. These issues can not only cause major inefficiencies but also cause the system as a whole to break down. These issues include systematic failures, which many computer (agent) systems suffer from.

- An inherent problem is that Contract Net assumes that all agents are friendly and benevolent. As a result, there is no mechanism to detect conflicts and more importantly solve them. Since each agent is self-interested, in reality it will likely be proactive and antagonistic. Present CNP does not provide any mechanism for ensuring trust and reliability of communicating agents.
- The communications infrastructure isn’t completely reliable. Nodes and/or links could fail at random creating pockets of isolating agents. The system must have mechanisms to try to reroute communications in the case of a failure. If communications can’t be reestablished, the agents affected must be able recover. An agent could also go offline (die) for any number of reasons and the agents involved would have to act in the same manner as for a communications failure.
As the period of each time slice is decreased, the communications (messages) required increases dramatically. Since the communications infrastructure is already a limiting factor, increased messages could cause further bottlenecks. A small period would also require agents to process messages and make decisions quickly. Also, a small period could mean that the Initiator misses out on lower cost bids because certain Participant(s) were not able to respond quickly enough.

Deliveries must occur at exactly the right time for certain inputs, such as electricity, otherwise problems, such as power outages, could occur. These problems could cost millions of dollars in lost production, downtime and information loss. In order to ensure delivery times, all the agents would have to run on a synchronized clock. However, synchronizing the internal clocks of all the agents could prove to be very difficult.

Finally, in any wide-scale implementation of Contract Net, the agents will likely be developed by many different parties. As a result agents can’t always be trusted to fulfill their end of a contract because of bugs and/or self-interest among other things. The system must have safeguards to deal with these problems and take preventive action if necessary.

2.9 Conclusions

The chapter presented various communication protocols deployed for agent communication in multiagent system. Also, since the main focus of current research work is contract net interaction protocol, therefore the same was presented in detail elaborating the architecture, working, challenges and its applications. Next chapter presents a detailed literature review highlighting the work of eminent researchers in the related domain and thereby analytically analysis the need of current work.