considered between the layers. It is much simpler than the horizontally layered architecture. But, this simplicity in the vertical architectures comes at the cost of decrease in flexibility i.e. in order to take a decision the control has to pass through different layers. This architecture is also not fault tolerant; the failure of process at any layer is most likely to have serious effects on the performance of an agent.

3.13 Conclusion

In this chapter, a comprehensive survey of the existing channel allocation schemes has been done. The chapter also discussed various agent architectures that are used to solve different real-time problems. The next chapter presents an Optimized Blocking Dropping Load Balancing (OBDBLB) algorithm.

Chapter 4
Optimized Blocking Dropping Load Balancing (OBDLB) Algorithm

4.1 Introduction

This chapter builds a specialized model based on hybrid agents and proposes an Optimized Blocking Dropping Load Balancing (OBDLB) algorithm for efficient channel allocation.

4.2 INTERRAP: Vertical Layered Architecture

INTERRAP is a technique to model resource-bounded interacting agents by associating reactivity with deliberation and cooperation capabilities [123]. INTERRAP is based on a BDI architecture, i.e., the informational, motivational, and deliberative state of an agent [9], which is explained through beliefs, goals, a rather generalized version of plans, and
intentions. Agent input (perception) is connected to agent output (action) through a set of operations that describe the inter-relationship between different mental categories of an agent. However, as compared to BDI architectures like [10], [22], the mental categories and the operational relationships between them are derived into three hierarchical layers and link them through a hierarchical control mechanism.

The development of INTERRAP agent architecture has been guided by the following common design decisions:

- **Layered control**: An agent is identified by different levels of abstraction, and of inferential and representational complexities.
- **Layered knowledge base**: The knowledge base is used to store the beliefs of an agent, and it allows to control the amount of information available to the lower control layers.
- **Bottom-up activation**: Control is moved bottom-up; layer $i$ takes control only if layer $i-1$ is not capable to deal with the situation.
- **Top-down execution**: All the layers use operational primitives specified by the
INTERRAP is an example of vertical layered architecture, in which control is activated by the behavior-based layer (BBL), which has direct access to the sensory system of an agent, and is moved upward incrementally until an appropriate layer capable for execution has been found. From there, the control moves back downward to the BBL, which is the only layer having direct access to the actors functions described in the agent's world interface. This is a major difference to other layered techniques, such as those by Ferguson [159] and Kaelbling [114] where there is simultaneous access to both sensory input and actors output, and the conflicts between the layers are resolved by applying proper universal filtering and suppression techniques, so that a particular layer only sees those parts of the input that are appropriate to it, and to suppress unsafe interactions between different layers.

Each control layer in an INTERRAP architecture is allowed to access only a particular portion of the agent knowledge base (KB); this access is organized in such a way that each control layer can use the information which is stored in the corresponding KB layer, and in the lower KB layers, but it is not permitted to use information stored in higher layers, e.g., the BBL can access only to the world model part of the KB storing the object-level beliefs of an agent, about the world. Whereas, the local planning layer can additionally use the mental model where information related to planning is stored. Finally, the cooperative planning layer may access the entire KB including the agent's
social model, consisting of negotiation protocols, and information related to the goals of other agents.

4.2.1 The World Interface

The world interface of an agent has three subsystems implementing agent's essential services for sensing its environment, executing actions, and for managing the communication with other agents. Each subsystem of the world interface is briefly discussed in the following:

- **The sensor subsystem**: A sensor corresponds to an object specifying methods for sensor calibration, for enabling and disabling its activity, and for reading the current value of an agent.

- **The actor subsystem**: The actor subsystem manages the physical actions the agent may take, e.g., robot’s motor control routines.

- **The communication subsystem**: The communication subsystem of a world interface specifies the technique of sending messages to and receiving messages from different agents. According to speech act theory, sending messages to other agents is viewed as a particular type of acting. Whereas, receiving messages are implemented using a sensing process modeling of the agent's receive-queue.

At the world interface, messages are represented as tuples:

\[
Msg = (Id, Sdr, Recp, Ref, Type, Content)
\]

where Id is a unique message identifier, Sdr and Recp are the sender and receiver respectively, Ref is a reference to a message-id, Type consists of one of a list of message types, and Content specifies the real message content.

Both received messages and perceived information are passed on to the agent knowledge base. The assumption of symbolic perception allows us to execute this by just asserting the corresponding information into the KB. Therefore, sensory information can be accessed either by the agent control unit or by the automatic update of knowledge base by
the world interface module. In addition, the world interface might get commands to send or receive messages and to activate actors.

4.2.2  **The Knowledge Base**

The knowledge base of an INTERRAP agent architecture is divided into three layers: (1) world model for reacting, i.e., stores the information about the world of the agent inhabits, (2) mental model for local planning, and contains the information about the agent's plans, (3) social model for cooperative planning, has the social knowledge about the other agents and their interactions. All the layers are activated bottom-up. If the lower layer fails to cope with a given situation, then the higher layer is activated.

4.2.3  **The Control Unit**

The control unit of INTERRAP agent architecture has three layers, i.e., behavior-based layer, local planning layer, and cooperative planning layer, which are explained as:

a. **The Behavior-Based Layer**

The behavior-based layer (BBL) provides two functions: firstly, it includes the reactive behavior of the agent, which permits it to deal in real-time during emergency situations. Secondly, it provides the procedural knowledge about the agent, which is required to perform routine tasks efficiently. The BBL communicates with the local planning layer by activating it and executing procedure calls present in the local planning layer, if situations arise that cannot be dealt with by the BBL itself. The behavior-based layer has to identify emergency situations and to take actions swiftly to these situations; it also needs to plan the execution of reasonably complex procedures. The main ideas and concepts for the behavior-based layer are the following:
• The section of the KB accessible to the BBL is the world model; this means that situation identification can be performed on a set of ground details, and efficient matching algorithms are present for this.

• The number of situations to be addressed is restricted by the set of situation descriptions that have to be supervised by the behavior-based layer. This, along with the propositional knowledge representation, allows an efficient accomplishment of situation recognition.

• There is a strong relationship between reactor goals and situations identified on the world model; identifying a situation directly resulted in the emergence of a reactor goal. Therefore, goal activation must be done efficiently.

• The mapping of intentions to the situation goals is achieved through patterns of behavior (PoBs), that provide a strong link between the executable programs and situations. Therefore, planning is only limited to situation-driven decision-making with a short look-ahead.

• Particular situation-goal pairs that are identified in the situation recognition part of the behavior-based layer, but are not matched through a reactor pattern creates an upward activation request to the local planning layer.

• It is the commitment towards the execution of procedures in the local planning layer that caused the activation of PoBs at the behavior-based layer.

Interfaces
This section explains how behavior-based layer behaves when it is interfaced with three other modules within the architecture: (i) the knowledge-base, (ii) the world interface, and (iii) the local planning layer:

Knowledge Base Access: The lowest part of the knowledge base (i.e., the world model) is accessed by the behavior-based layer, where accurate information regarding the environment is stored. Firstly, the basic data structures such as the descriptions of PoBs and organizational information about the PoBs to be monitored, are stored inside the agent knowledge base. Secondly, situation recognition is done over the world model utilizing the basic knowledge access functions. Thirdly, the belief abstraction is done by the control layers.
Connection to the World Interface: In the context of granularity, the behavior-based layer calls actors and sensor functions explained in the world interface. In addition to that, it requests access to the content of the message queue and current sensory values. This information is stored in the perception buffer of the agent. By activating the suitable update routines, the recent information is provided in the knowledge base.

b. The Local Planning Layer (LPL)

Planning systems are also called computational systems because, for a given set of goals, it produces a sequence of actions which will lead to the accomplishment of the goal or task. The design of a planning system is one of the oldest fields of Artificial Intelligence. The local planning layer contains the services of an agent to develop plans for accomplishing its local goals and its local tasks. Architecture for autonomous but interacting agents must support the programmers in programming goal-directed agents. It should also permit the designers of agents to define actions by pre-conditions and effects, a planning mechanism can compute a plan after admitting instances of planning problems. However, due to the absence of powerful and good general planning mechanism, INTERRAP general agent architecture, face the following problems: if we trust a particular planning mechanism at the local planning layer, it might generate good results for some applications, but insufficient for others. Whereas, if no such commitment is made at all, then the designer of a system has to program its own planner for each application.

A compromise solution has been suggested in order to solve this problem, by restricting the potential planning mechanism from two sides: (i) every planning method to be used in INTERRAP should provide proper interfaces to its neighboring layers. (ii) the control process of the local planning layer must determine how the layer works. One step inside this control process is to call a plan generation function, when a description of the planning problem is given as input. Thus, the fundamental idea is to present an open architectural framework depending on the domain under consideration. This idea is not perfect, and has many restrictions e.g., different
planners need different descriptions of the available actions, or they need exact descriptions of the knowledge. However, these problems can often be solved by providing suitable transformation functions.

Fig. 4.2 shows the functional structure of LPL. It contains several functional modules: (i) the controller, (ii) the plan generator, (iii) the plan interpreter, (iv) the plan evaluator, (v) the plan scheduler, and (vi) the plan executor. The controller runs the control cycle for the local planning layer, and it also acts as an interface to the neighboring layers. The actual planning mechanism is contained in the plan generator, and also has access to a plan library storing pre-defined plans. The plan interpreter keeps a set of goals. The plan evaluator calculates the usefulness of these plans, which is used to choose a best plan for a particular case. The plan scheduler uses the output of a plan interpreter to create a schedule. A schedule is a sequential ordering of plan steps. Finally, the timely monitoring of the schedule and activating the execution of scheduled tasks is done by the plan.
Fig. 4.2: Functional Structure of Local Planning Layer

Interfaces
The local planning layer in the INTERRAP agent architecture interacts with three other modules: (i) the knowledge base, (ii) the behavior-based layer, and (iii) the cooperative planning layer. The interfacing of local planning layer with the other modules is explained below:

Knowledge Base Access: The local planning layer not only contains access to the world model but it can also access the mental model of the knowledge base where goals, plans, schedules, and plan libraries are stored. In the INTERRAP agent architecture, the knowledge defining the mental model of an agent is managed locally inside the control layer and access to this information does not need physical access to the knowledge base. Therefore, concept of a mental model inside the knowledge base is purely conceptual rather than physical.

Interplay with the Behavior-based Layer: There are two main modes of interaction of local planning layer with the behavior based layer: (i) the upward activation requests are received by the local planning layer from the lower layer; and (ii) commitments for executing PoBs are sent downward to the behavior based layer. Messages which are passed on to the BBL are created by the planning mechanism.

Local and Cooperative Planning: The interface between the LPL and cooperative planning layer (CPL) explains how to co-ordinate a local task planning with multi-agent planning: Firstly, whenever a LPL fails to solve a problem satisfactorily, then an upward activation request to the CPL is generated. Secondly, commitments for executing a single-agent plan for the LPL are received from the CPL because of the cooperative planning process.

c. The Cooperative Planning Layer
Autonomous agents that are present in the multi-agent environments must cooperate and coordinate their actions with other agents. Negotiation is the important tool for coordination; and communication is the basic requirement for negotiation. The cooperative planning layer provides universal negotiation mechanism and allowing
the agents to participate and agree on common goals, to assign tasks, to settle conflicts, and to synchronize their local plans.

The cooperative layer contains tools for identifying interacting situations and deriving goals out of these situations. It also provides functions to select strategies for negotiation with other agents. In this research work, the assumption is that different interacting agents negotiate regarding joint plans that allow them to achieve individual or common goals. The structure of cooperative planning layer is shown in Fig. 4.3. It functions in a control cycle which receives requests from the local planning layer, additional information is collected, and then, an appropriate protocol and negotiation method is selected. The planning, scheduling, and execution of tasks of the CPL contain the processing of protocols based on negotiation methods. During protocol execution, the agent must gather sufficient information about the goals of other agents to be able to categorize the interaction situation and to create the input explanation of a cooperative planning problem. The solution for such a planning problem is an appropriate joint plan.

In the cooperative planning layer, planning, scheduling, and execution of tasks take place at two different levels: at the meta level, and at the object level. At the meta level, processing of protocols is done according to strategies, whereas at the object level, plan-generating functions are called from inside the protocols in order to create the negotiation set.
Fig. 4.3: The Cooperative Planning Layer

**Interfaces**

The cooperative planning layer is interfaced with both the local planning layer and with the knowledge base. The association between the cooperative planning layer and local planning layer is already discussed in the previous section.

All layers of the agent knowledge base are accessed by the cooperative planning layer, where the information present in the world model is accessed in order to know the current state of the world. The information regarding the agent’s goals is accessed from the mental model. The cooperative planning layer is activated by the lower layers (i.e. bottom-up), and the layer gets the necessary information about the current state of an agent from the local planning layer. Therefore, cooperative planning layer does not obtain the specific knowledge about the conflict situations, and the situations are classified by their mental and external context. At the cooperative planning layer, the information about the goals of other agents improves this explanation of an interaction situation, and this information is supplied by the social model. On the basis of this information, the interacting situations are then classified by the cooperative planning layer. The internal structure of each layer of INTERRAP agent architecture is explained in the following section.
Let $B$, $G$, and $I$ be the beliefs, goals, and intentions of an agent and $P$ represent a group of perceived propositions. The INTERRAP agent architecture implements the following three basic functions:

- $\text{BR}(P, B) = B'$ is a knowledge abstraction and belief revision function, the current perception $P$ of an agent and its old belief $B$ is mapped into a group of new beliefs $B'$.
- $\text{SG}(B, G) = G'$ is a goal activation and situation recognition function, where new $G'$ goals are derived from the agent's beliefs $B$ and its existing goals $G$.
- $\text{PS}(B, G, I) = I'$ is a planning and scheduling function, which is used for deriving a new set intentions $I'$ based on the current intentions $I$, the beliefs $B$, and the goals $G$ of the agent.

Table 4.1 shows how SG, PS, and BR are distributed over different control layers of INTERRAP.

<table>
<thead>
<tr>
<th>Layer Function</th>
<th>BBL</th>
<th>LPL</th>
<th>CPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>Generation and revision of beliefs (world model)</td>
<td>Abstraction of local beliefs (Mental Model)</td>
<td>Maintaining models of other agents (social model)</td>
</tr>
<tr>
<td>SG</td>
<td>Activation of reactor patterns</td>
<td>Recognition of situation requiring local planning</td>
<td>Recognition of situation cooperative local planning</td>
</tr>
<tr>
<td>PS</td>
<td>Reactor PoB: direct link from situations to action sequences</td>
<td>Modifying local intentions; local planning</td>
<td>Modifying joint intentions; cooperative planning</td>
</tr>
</tbody>
</table>

The processes performed at the different control layers of INTERRAP architecture have several similarities, and different instantiations of SG and PS are explained by them. Therefore, a uniform structure that is shared by each layer is presented based on
these observations. Fig. 4.4 shows the internal architecture of INTERRAP control layer. Each control layer $i$ of an INTERRAP architecture contains two processes utilizing functions SG and PS, and these function interact with other, as well as with processed present in the neighboring layers.

- The recognition of a specific situation and activation of goals for a particular control layer is done by the SG, i.e., situation recognition and goal activation function.

- The mapping of goals to intentions and to actions is implemented by the planning and scheduling function PS. The goal-situation pair produced by the SG component is received as input by the PS, the plans are determined to achieve these goals, the plans are scheduled and their execution in the agent structure is monitored.

---

**SG-Situation Recognition & Goal Activation Process**

**PS-Planing, Scheduling & Execution Process**

---

Information Flow
- Main control Flow
- Additional control flow
4.2.41 The Flow of Control

The behavior of an INTERRAP agent is analyzed from the interaction among individual control layers [63]. Bottom-up and top-down control are the two basic control directions. For dealing with different types of problems, three general control paths in an agent describing three types of control flows are determined. Next section discussed both top-down and bottom-up control flow, and illustrates the general paths of control through an INTERRAP agent.

Bottom-up Control: The beginning of activity in INTERRAP is managed in a bottom-up manner via the upward activation request mechanism [116]. The process SG\textsubscript{i} recognizes new situations at layer i, and then at this layer the planning, scheduling, and execution process PS\textsubscript{i} is activated. If PS\textsubscript{i} decides not to be capable enough for handling the corresponding situation-goal pair, then an activation request is sent to the situation recognition process SG\textsubscript{i+1} at layer i + 1; there, the description provided by layer i is improved by extra information which is available to layer i + 1 and that is essential to generate an appropriate goal description for PS\textsubscript{i+1}. Finally, the result of the activation request is reported to PS\textsubscript{i}.

A competence-based control flow is implemented by this mechanism, which make sure that situations expecting swift reaction are managed by the behavior-based layer, whereas other situations which encode complex planning problems are moved upward until a layer which is capable for solving the problem has been reached. The benefit of this approach is that, it s not necessary for the lower layers to know the capabilities of higher layers, and they only have to decide whether they are capable to solve a particular problem for themselves or not. This can be done quickly and efficiently at the behavior-based layer based on simple matching and table lookup mechanisms.
**Top-down Control:** In the top-down control flow, activation of control layers is performed bottom-up, but the acting is performed using a top-down mode. The neighboring layers coordinate the activities of their planning and scheduling processes through communicating commitments, and the direction of communication is from higher to lower layers. Commitments passed on to the local planning layer by cooperative planning layer are the partial plans that are invented by the CPL during a joint plan negotiation, and explain agent’s role in a joint plan. These commitments are included into the schedule managed by the local planning layer. Whereas, the local planning layer is completely dedicated to the execution of procedure PoBs, and these commitments are passed to the behavior-based layer which, in turn, activates the corresponding procedures. The commitments present at the behavior-based layer cause the execution of functions at the agent’s world interface. The layer i, after passing a commitment to layer i-1, wait for an acknowledgement to determine whether the commitment has been executed successfully or not. Based on these reports, the higher layer decides whether to continue executing its plan or has to re-plan.

**Generic paths of control:** In bottom-up control, the activity is always started by the behavior-based layer [202], and there are three basic control paths that start from there, as shown in Fig. 4.5(a-c). Fig. (4.5.a) illustrates the reactive path: situations that can be dealt with by routine PoBs without doing any explicit planning are identified and managed at the behavior-based layer. If there exits no suitable PoB for complex situations, then the control follows the local planning path. The situations that are managed by this path require local task planning. Then the control is moved upward to the local planning layer, where the joint plan is negotiated, and the planning process is driven by the PoBs. Fig.
Fig. 4.5: Generic Control Paths

Often, the activities taking place at the different layers of an INTERRAP architecture are neither clearly separated nor they follow the severe temporal ordering as shown in the Fig. (4.5.a-c). Rather, the control swaps between different layers during processing. Fig. (4.5.d) shows the non-contiguous arrangement of planning and execution; planning is a continuing incremental process, and the result of present PoB calls affect the future planning decisions. Analogously, Fig. (4.5.e) shows that there is a close relationship between cooperative and local problem solving.

According to Ferber [63], Labidi et al. [116], and Wooldridge et al. [202], agents are of two types i.e., reactive and cognitive. Reactive agent architecture was developed by Brooks [28] for controlling autonomous mobile robots, as he was very much against cognitive agents. Reactive agent architecture is set of task executing behaviors (Jennings et al., [97]), and every task executing behavior is a finite state machine in which an action output is continuously mapped to the perceptual input. This mapping is accomplished through a set of rules, which decide about the action to be performed, and the decision is based on the agent’s current state.

Several hypothetical agent architectures have been proposed to explain the basic model of cognitive agents. According to Labidi and Lejouad [116], Khoualdi [113], Noubissie
Tchako [143], and Wooldridge [203] the internal architecture of agent posses the following properties:

- **Perception:** The perception module is one of an agent’s interfaces to its environment. Commonly the perception module obtains signals from the agent’s sensors. But in most of the architectures, this module is integrated into the communication interface.

- **Self-knowledge:** The agent’s self-knowledge contains agent’s knowledge about itself, including its physical state, location and skills, etc.

- **Domain knowledge:** This knowledge concerns the problem-solving domain and environment. Usually this module contains the description of the problems to be solved.

- **Social knowledge (acquaintance knowledge):** This knowledge, also called beliefs, is the knowledge used by the agent to interact with its acquaintances. It describes the skills and identity of acquaintance agents. The knowledge is used by the agents to identify other agents with whom it is useful to interact, and wish to determine which agents have the skills necessary to perform a particular task. This knowledge must indeed model the role, competence, the localisation (address of an agent), the goals, the plans, and the resources of these dealings to be able to interact with them.

- **Learning:** An agent working in a dynamic environment needs to adapt to changes in that environment. It needs to learn in order to update its knowledge about its environment, other agents, and the problems to be solved.

- **Reasoning:** It is the decision making process which decides to act on the basis of the information it receives, and in accordance with its own objectives to achieve its goals.

- **Communication:** It is the interface used by the agent to communicate with its environment and with other agents.

- **Cooperation:** Defines the models of coordination and cooperation to interact with other agents in order to perform tasks for other agents.

### 4.3 Multi-Agent System
According to Durfee et al. [145], MAS is a loosely coupled system of problem solvers that work collectively to solve problems which are beyond the individual capacity of each of the problem solver. Another definition was proposed by Ferber [69], and according to him MAS as a system consist of collection of autonomous agents, which communicate with each other to accomplish common objectives, while at the same time, each agent pursue its individual objectives. Fig. 4.6 shows the MAS.
The major characteristics of MAS, as explained by Jennings et al. [97], are:

- For solving any particular problem, each agent does not have complete information and capability; therefore it has a limited viewpoint;
- Having no global system control;
- Data is not centralized;
- Having asynchronous computation

For multi-agent systems to solve common problems, the agents must coordinate, cooperate, and communicate with each other. Agents are required to communicate with other agents to accomplish their objectives either they do not have enough capability to solve the problem alone, or the agents are interdependent on each other. The interaction or communication between the agents can vary from simple information exchange to request for specific action to perform (Jennings et al., [98]).

a. **Cooperation and Coordination**

Durfee et al. [44] started a work on cooperative distributed problem solving where they explained how a loosely coupled network of problem solvers can interact with each other to solve problems that may be beyond the scope of their individual capability. Each problem-solving node i.e. agent in the network is only able to solve sophisticated problems and can work separately, but the problems faced by the agents or nodes cannot be finished without cooperation. Cooperation is essential because none of the node has sufficient capability, information, and resources to solve a problem, and various nodes may have skill for solving different sections of the problem.

- **Distributed Expertise, Resources, and Information:** To solve the problem no single agent has adequate capability, resources, or information. Individual agents working in separation fail to solve several practical problems because they do not have the required skill, resources, or information. It is required to combine different skills, in order to solve difficult problems which are out of the capacity of any single agent. Different agents may schedule different resources to produce
a final product. Finally, various agents in a system may have different viewpoints of a problem, and agents can also have different capabilities and expertise.

- **Preventing chaos:** Coordination is very important because the decentralization in the agent systems can bring in chaos very easily. No agent has the global view of the entire system to which it belongs, as it is not feasible to have realistic complexity. Therefore, agents only have local views, goals, and knowledge that can clash with others.

- **Interdependencies between agents:** Interdependence between the agents happens when goals set by individual agents are linked either because local decisions taken by one agent influence the decisions of other community members, or due to the likelihood of conflict between agents.

- **Efficiency:** Cooperation and coordination between the agents can drastically increase the efficiency. Even though agents can function separately, thereby eliminating the requirement for cooperation, but the information found by one agent can be useful for the another agent, and both the agents can jointly solve the problem twice as fast.

The major issues addressed in the inter-agent cooperation are as following [203]:

- Collaboration by task allocation or distribution: how tasks are allocated and distributed between agents?

- How to coordinate activities in order to avoid conflicting situations (coordination)?

In the next section we are going to discuss some techniques that are developed by the MAS community to address the abovementioned concerns.

i. **Collaboration by Task Allocation**
Task allocation includes the organizational mechanisms through which the agents can come together perform tasks collectively. The allocation of tasks can be controlled by centralizing the entire distribution process ([145], [70], [176]).

**ii. Task Allocation Through Mediation**

In centralized task allocation, special agents (mediators or traders) are used to control the allocation of tasks to the agents. The mediator agent in this case must have the required knowledge of all the agents present in the system including their capability and ability. Fig. 4.7 shows how a mediator agent is allocating a task T to an appropriate agent. When a request is received by the mediator agent to carry out the task, it then forwards the request to relevant agents. If any of the agents becomes ready to carry out the task, then the mediator agent allocates the task to that particular agent. If more than one agent is ready to accept the task, then the mediator selects the one based on some criteria. If all the agents refuse to accept the task, then the originator of the request is informed by the mediator that it failed to find a suitable agent for the task.

![Fig. 4.7: Centralized Allocation of Tasks](image-url)
iii. Distributed Task Allocation Through the Contract Net Protocol

In distributed task allocation, every agent independently finds the appropriate agents that are capable to carry out its task without any centralization. A contract net protocol (CNP) is a well known example of distributed task allocation. In agent based systems CNP is the widespread and best-studied method for distributed task allocation ([176], [203]), and it is the high level protocol for accomplishing efficient cooperation ([183]). As the name of the protocol suggests, the basic metaphor used in CNP is contracting. The CNP got an excited reception from the distributed artificial intelligence community.

A decentralized market organization is assumed in this approach and agents can accept two roles: a manager and a contractor. The basic idea of this type of cooperation is that if an agent fails to solve an allocated task using local resources; it will try to find other enthusiastic agents with the required resources in order to solve the tasks, and the contracting scheme is used to allocate the tasks. The tasks are advertised by the manager agent using the task announcement to other agents present in the system, as shown in the Fig. 4.8. The contracting agents in response to the announcement assess the tasks with respect to their own capability and submit the bids. The submitted bids are evaluated by the manager, and select the most suitable bidder to perform the task, and then the contract is awarded to the contractor with the most suitable bids. The contractor then takes the responsibility for the execution of the tasks, and after the task is finished, a report is sent to the manager.

In the CNP, the manager has to wait for all the bids before it starts evaluating them, this can actually cause the manager to wait for bids indefinitely. In order to overcome this problem, a deadline is assigned to each task announcement for the receipt of bids, and all the bids received after this deadline are rejected [70]. Another problem associated with CNP is that between the time a proposal is submitted by the bidder to its manager and the time the contract is awarded or rejected, it is committed to carry out the task. Therefore, to solve this problem, the CNP is extended to a time bound
negotiation framework by linking the commitment duration to the bid messages and task announcements.

Contracts have been binding in the basic CNP, i.e. if an agent agrees to a contract; the contract should be honored by it with full commitment. Even though a contract may be beneficial for an agent when it is established, it may not be beneficial after some future events have taken place. In most of the realistic scenarios, agents are best suited in dynamic environments, where agents might be aware of new information, and another agent may try to communicate with it. In response to these situations, the CNP has been extended to leveled commitment contracts, which is another method of linking commitments to the negotiation protocol ([95], [172], [96], [18]).

![Contract Net Protocol](image-url)

Fig. 4.8: Contract Net Protocol
b. Coordination

Agents communicate with each other in order to attain better goals of themselves or of society in which they are present. Communication allows the agents to coordinate their behavior and actions, resulting in a more consistent system. The degree of coordination is the amount to which the irrelevant activity can be avoided by minimizing resource contention, avoiding deadlock and livestock, and keeping appropriate safety conditions. Cooperation is the coordination between non-antagonistic agents, whereas negotiation is the coordination between competitive agents. In order to cooperate successfully, every agent should keep a model of the other agents, and also build up a model of future communications or interactions. This presupposes sociability.

The major techniques that have been developed for coordinating actions are centralized planning, negotiation, and multi-agent planning.

4.4 Functional Specification of Hybrid Agent Architecture for Solving Channel Allocation Problem

For solving channel allocation problem, the functional specification of the hybrid agent architecture (INTERRAP) is presented below:

4.4.1 Behavior–Based or Reactive Layer

The job of a reactive layer is to allocate channels to the new as well as to the handoff calls. The basic channel allocation algorithm used in this layer for allocating channels is explained as follows:

4.4.1(a) Hybrid Channel Allocation Algorithm

The selection of a particular hybrid channel allocation algorithms is little bit tough because of cost and which channel is appropriate for the user. The major purpose of all the algorithms is to assign a cost function for allocating each of possible candidate
channels, and then choose one channel with a minimum cost. The calculation of the cost function is based on one or more of the following aspects; future call blocking probability; re-use distance calculation; channel occupancy distribution; radio signal quality measurements etc.

4.4.1(b) **Proposed Algorithm : Optimized Blocking Dropping Load Balancing (OBDLB) Algorithm**

To achieve the objectives of research work, a hybrid channel allocation algorithm Optimized Blocking Dropping Load Balancing (OBDLB) is proposed. According to this algorithm, when a new call request or hand-off request is received on the system, the system always gives priority to the hand-off call than a new call. When call is received, the system first checks from where the call is originated, then system will confirm the cell ID. Synchronization channel is used to check the cell identity, which carries the information of the Base Station Identity Code (BSIC), synchronization channel is transmitted by the Broadcast Control Channel (BCCH). After receiving the cell ID, the system then checks the list of channels that are currently in use in that particular cell where the call is originated, and synchronization channel (SCH) is used to provide the information about the channels present in that cell. The SCH then carries the information to enable the mobile device to synchronize to the TDMA frame structure and know the timing of the individual timeslots. After receiving the call, the system will first search for a free channel present in that cell where the call is originated. If free channel is found in the cell, it will allocate a free time slot to the call. If there are no free channels available in the cell then the OBDLB algorithm will search for a new channel in the poll based on the cost function. Channel which satisfies the cost is allocated to the cell. Before searching a new channel for the call in the poll, the algorithm will check the number of channels which are in use in that cell. If number of channels present in the cell is more than the threshold of 25% then the system will not assign any more channels to the cell and the call will be blocked or dropped. OBDLB algorithm selects a random free channel from the poll which satisfies the interference constraints. The cost (interference) is calculated using the following aspect:
take the first available channel from the pool and compare it with the channels already in use in that cell as well as in the neighboring cell.

All the above information will be provided by BCCH; BCCH carrier carries the information about neighboring cells, which are monitored by the mobile, cell identity, and the list of frequencies used in the cell. After getting all the information, system will compare that first available channel from the pool with the list of channels are in-use in that cell and also with neighboring cells. If any of the conditions is not met then the system will not assign that channel for a call, because interference is the cost limiting factor. The channel that fulfills the above given requirement will be assigned for the call. If all of the channels are not found suitable then the call will be blocked or dropped. Various steps that are involved in allocating channels to the calls are shown below:

**Step 1. Call is received**

**Step 2. System first checks whether it is a new call or a handoff call**

**Step 3. If it is a handoff call**

a. System first identifies the cell-Id or a sector-Id of the cell or the sector where the call is coming from

b. After identifying the cell or sector Id, the system then moves on allocating the free channel to the call

c. The search for the free channels begins with the nominal channels first that are allocated to the cell and it then moves to step 1

d. If free nominal channel is not found then search process shifts to the dynamic channels present in the cell and it then moves to step 1

e. If all the dynamic channels are also busy then the free channel is searched from a pool and then it moves to step 2.

**Step 4. If received call is a new call**

a. After identifying the cell or sector Id, the system then moves on allocating the free channel to the call
b. The search for the free channels begins with the dynamic channels that are allocated to the cell and it then moves to step 1.

c. If free nominal channel is not found then search process shifts to the dynamic channels present in the cell and then it moves to step X.

d. If all the dynamic channels are also busy then the free channel is searched from a pool and it moves to step Y.

**Step X:**
1. Select a first frequency channel from a cell
2. Find a free time slot in the selected frequency
3. If a free time slot is found then allocate it to the call
4. If free time slot is not found in the selected frequency then move to next frequency
5. Search all the frequencies to find a free time slot

**Step Y:**
1. Select one frequency channel from the poll
2. Before allocating a frequency channel to the call, compare if number of channels present in the cell ≥ threshold level of 25% (a cell cannot hold more than 25% of the total channels allocated to the network)
3. If number of channels present in the cell is more than the threshold value then no more channel would be allocated to the call and the call would either be blocked or dropped.
4. If number of channels present in the cell is less than the threshold value then channel is selected from a poll for allocation to the cell
5. Channel is allocated to the cell if it satisfies the cost function. The cost is based on the interference constraints, current degree of coldness.
6. If no channel is found from the pool that satisfy the cost function then call would be dropped or blocked.

4.4.2 **Local Planning Layer**
Local Planning performs channel reassignment based on signal-to-noise ratio calculations. When a departing user releases the channel, the reactive layer performs the following re-assignment decisions:

- If it is a dynamic channel, it will either be allocated to a new call or given back to the pool, depending upon the traffic load in that particular cell
- If it is a nominal channel, it will be re-assigned to a new user or non-departing user.

4.4.3 Co-operative Planning Layer

The Co-operative planning layer performs the load balancing of the entire network in order to keep the low rates of call blocking and droppings by moving the calls from heavily loaded cells, also called hot cells to less loaded cells or regions. This process is called deliberate traffic handoff. Calls of the mobile users which are close to the cell boundaries are only moved to the neighboring cells. The handoff must be performed in a co-ordinated manner in order to avoid mobile users being shifted back and forward between various cells. It is basically a load balancing problem and various agents are carefully selected to engage them in a joint plan to take the handoff decisions.

A Contract Net Protocol (CNP) is used to engage the agents for communication. Contract Net Protocol specifies the interaction between agents for fully automated competitive negotiations through the use of contracts. The interaction process in the Contract Net protocol has the following four steps:

1. The Initiator sends a Call for Proposals (CFP) to various Participants.
2. Each Participant reviews Call for Proposals 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.
The above four steps implemented in this study are as follows:

1. The agent $A$ in the hot cell also called the Initiator, sends out a Call for Proposal $CFP-1$ to various Participant agents $A_i$ in the co-channel cells, where $1 \leq i \leq 6$.
2. The $A_i$ after receiving the request sends out $CFP-2$ to their neighboring cell agents $B_{ij}$, where $1 \leq j \leq 6$, for channel availability assessment.
3. The $B_{ij}$ agent sends back to $A_i$ agent either $propose-2$ if it is ready to engage in a joint plan or $refuse-2$ if it is busy. The propose includes degree of coldness of a particular cell.
4. Each $A_i$ agent after receiving the answers of its neighbouring $B_{ij}$ agents computes the value of utility function based on the degree of the coldness of all the $B_{ij}$.
5. The agents $A_i$ that are able to perform the calculation based on degree of coldness sends the result to agent $A$ in $propose-1$.
6. The agent $A$ after receiving results from all the co-channel neighbours selects the one with biggest value for moving the calls. Then agent $A$ informs the selected co-channel cell also called winning agent with the $accepted-proposal-1$.
7. The winning co-channel cell agent then assess the availability of each $B_{ij}$ agent by sending $cfp-3$ to engage them in a joint plan, considering the number of plans it is already engaged in. The agent replies positively with $propose-3$.
8. Then the initiator sends an $accepted-proposal-2$ and $accepted-proposal-3$ to selected $B_{ij}$ cell.
9. After receiving back replies from its neighbouring agents, the winning co-channel cell agent informs the agent $A$ whether to accept with proposal with $inform:jp$ or reject its proposal with $failure:jp$.
10. The agent $A$ sends $inform:activejp$ to inform all the partner agents that joint plan is started.
12. The agent $A$ then sends a $query-ref-1$ to all partner agents.
13. Each partner agent replies with $inform-ref-1$ which includes total number of channels present and numbers of channels busy in the agent.
14. The Initiator agent $A$ computes the rate of change of channel occupancy $\Delta C_i$
\[ \Delta C_i = (\text{Avg. channel occupancy in cell } i) - (\text{Avg. channel occupancy in all the cells of joint plan } i) \]

15. Based on calculations of \( \Delta C_i \) the Initiator A informs the cell \( i \), whether to move the calls to neighbouring cells or not in order to balance the load.
4.5 Conclusion

Intelligent agents are the emerging subfield of artificial intelligence as they are important users of AI techniques. As agents can be considered as an end application of AI, the concepts of intelligent systems and fundamental architectures are introduced by their study. An introduction to agent and multi-agent systems is provided in this chapter. Different multi-agent architectures e.g. reactive, deliberative, and INTERRAP etc. are also explained. This chapter described the INTERRAP multi-agent architecture and new hybrid channel allocation algorithm OBDLB for the efficient allocation of radio channels. The major functionality of each layer of INTERRAP agent architecture was also explained. An INTERRAP multi-agent architecture consists of a knowledge base, world interface, and three control layers: (i) the reactive or behavior based layer performs the task of allocating channels to the calls; (ii) channel reassignment based on S/N ratio calculations is done by the local planning layer; (iii) the co-operative planning layer performs the load balancing of the entire network.