Chapter 1

System Implementation

In this chapter, we review information on the implementation of the work. The Communication Protocols (i.e. inter-agent control) that describe the way that the agent is connected and the Agents Behavior (i.e. intra-agent control) that explain the internal operation of each agent are presented here in more detail. Also, we present the different algorithms we implement, and the Mechanism Design and Charging Station agents uses in their strategies.

For the implementation of agents' microservices we used the SYNAISTHISI IoT Platform. It allows agent to communicate using MQTT publish/ subscribe protocol that is a widespread connection method for IoT devices. SYNAISTHISI provides a friendly and comprehensive Python API for agent development and management. The services and sub-modules made in the platform come in docker containers, allowing scalable and operating system independent deployments.

The inter-agent and intra-agent control is illustrated using statechart as described in Unified Modeling Language (UML). The statechart are designed using the diagramming and vector graphics application Microsoft Visio.

1.1 Communication Protocols

The Communication Protocol (CP) is set of standard actions that agents use to communicate with each other.

In Figure 1.1 we show the connection of each agent and the protocols that uses to pass the necessary information through the MQTT topics.

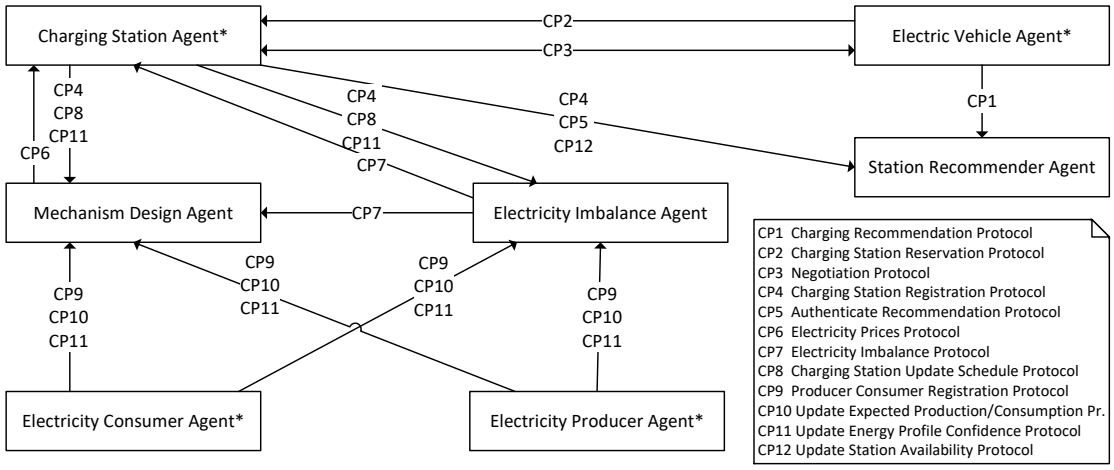

Figure 1.1: Overview of the proposed architecture. Stars (*) denote agent types with multiple instances. Arrows start from the agent that initiates the protocol and point to the receiver agents.

For each protocol is presented a state diagram that illustrates the agents that participate, their actions, the messages they send and the topic that sent over.

The figures of the communication protocol are split into two parts. In the left part there is the initiatior of the protocol and in the right the responder. START-states show the beginning of the execution (circle). Basic states (shown with green color) are where agent activities are executed and END-states show where execution ends (black dots within a circle). Transitions from one state to the other occur *(i)* when the activity of the source state finishes and there is no event on the arrow, or *(ii)* when the event on the arrow takes place.

Next for each communication protocol, we provide a detailed description of the operation and a figure to show it graphically.

1.1.1 Charging Recommendation Protocol (CP1)

In CP1 the Electric Vehicle starts the protocol and enters the state *SendRecommendationRequest*. Electric Vehicle publishes a message on topic *"EV/+/RequestChargingRecommendations"* that contains an array with *preferences* and *location*. The plus sign (+) in the topic is replaced with the ID of Electric Vehicle that sends the message and it is used like MQTT single-level wildcard defines. Station Recommender receives the message and enters the *ReceiveRecommendationRequest* state where it confirms that the request is

valid and finds the ID of the Electric Vehicle. After that, Station Recommender enters the *CalculateChargingRecommendations* state that according to the Electric Vehicle's preferences, the location and the availability of the Charging Station that is informed finds the most suitable electrical chargers. Then, Station Recommender enters the *SendChargingRecommendations* state that sends a message on topic *"EV/+/ChargingRecommendations"* that contains a list with the most matched Charging Slot that are available. Electric Vehicle receives the message, enters the *ReceiveRecommendations* state, and finishes the protocol.

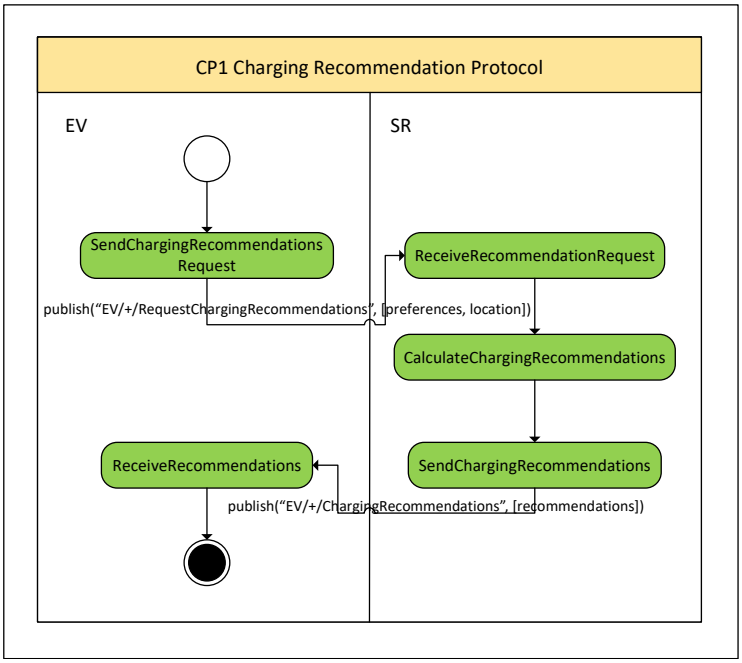

Figure 1.2: The model of the Charging Recommendation Protocol (CP1)

1.1.2 Charging Station Reservation Protocol (CP2)

After the Electric Vehicle agent completes the Charging Recommendation Protocol (CP1), a frequent sequence is the selection of a recommendation and the interaction with the corresponding Charging Station agent using the Charging Station Reservation Protocol (CP2).

The Electric Vehicle agent begins the protocol and enters the *SendStationReservationRequest* state where it publishes on the topic *"CS/+/ReserveChargingSlot"* (the plus sign is replaced with the ID of the addressed Charging Station) and a message that contains the chosen recommendation, the battery status, and information about its preferences. The Charging Station agent receives the message and enters the *ReceiveChargingReservationRequest*. Then enter the state *HandleChargingReservationRequest* where the agent

checks if the recommendation is genuine by using CP5, creates the reservation and, calculates the charging schedule according to the selected algorithm from 1.3.2. It goes on the *SendReservationOutcome* and publishes on topic *"EV/+/ReservationOutcome"*, replacing "+" with the Electric Vehicle's ID, the generated reservation, the schedule that represents the amount of energy the Electric Vehicle will charge or discharge each timestep and the price of buying and selling an amount of energy (i.e 1 kWh) each timestep. The Electric Vehicle receives the outcome message, enters the *ReceiveReservationOutcome* and terminates the protocol by entering the final state.

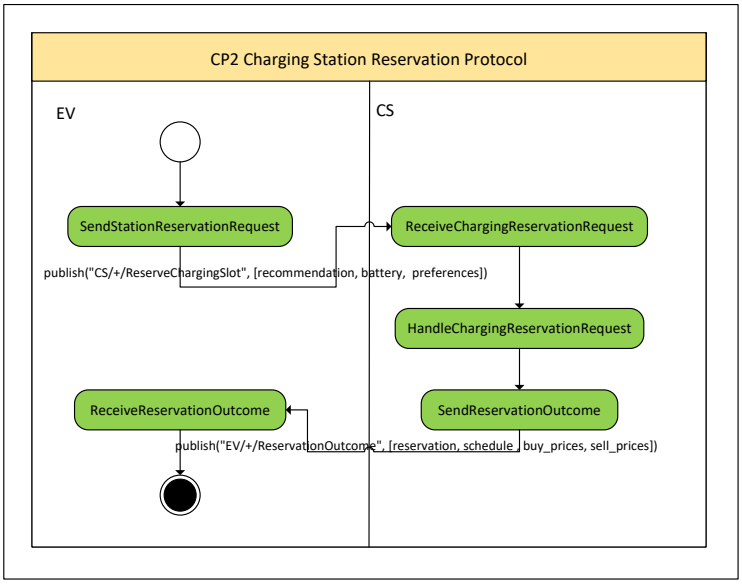

Figure 1.3: The model of the Charging Station Reservation Protocol (CP2)

1.1.3 Negotiation Protocol (CP3)

The Negotiation Protocol (CP3) is the communication protocol that enables the negotiation between the Electric Vehicle and Charging Station agents about a reservation that has already been agreed. There are two roles in the protocol, the initiator and the responder. This protocol is symmetrical that means that either the Electric Vehicle can initiate it and the Charging Station will be the responder or vice versa. The initiator starts the protocol, enters the *SendNegotiationMessage* state, and sends the message with *NegotiationObject*, which can be the arrival time, on the topic *"CS/+/EV/+/EV_start/start_negotiation"* if the Electric Vehicle is the initiator or on *"CS/+/EV/+/CS_start/start_negotiation"* otherwise. The first plus sign is replaced with the ID of the Charging Station agent and the second with the ID of the Electric Vehicle agent. The responder receives the message that is published and enters the *ReceiveNegotiationMessage* state.

Follow the *NegotiationDecisionMaking* state where the decision process takes place and the responder decides if it accepts to modify its reservation according to the provided negotiation object. Then enters the state *SendDecisionMakingOutcome* and publishes on the topic *"CS/+/EV/+/EV_start/accept_negotiation"* or on *"CS/+/EV/+/CS_start/accept_negotiation"* the *Decision* that can be *"Accept"* or *"Reject"*. The initiator takes the outcome message, enters *ReceiveDecisionMakingOutcome* where it processes the decision, and goes to the final state that terminates the protocol.

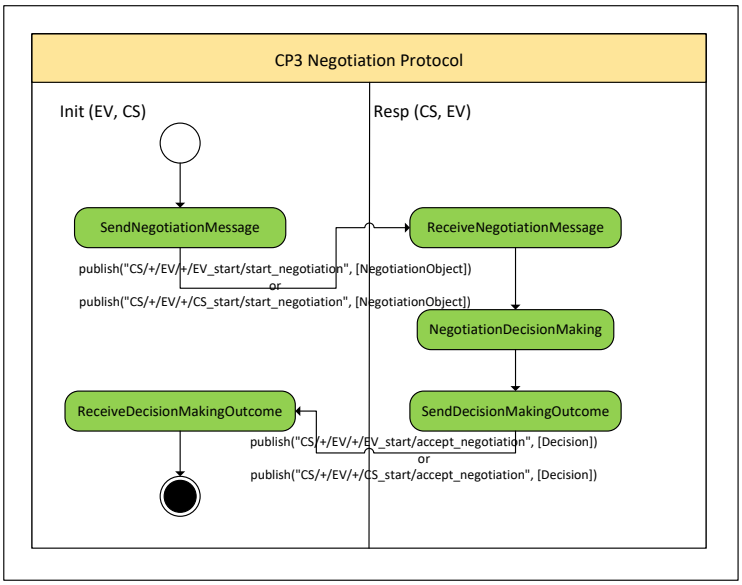

Figure 1.4: The model of the Negotiation Protocol (CP3)

1.1.4 Charging Station Registration Protocol (CP4)

In the Communication Protocol that manages the registration of Charging Station, there is a connection between the Charging Station and service providers that includes the Electricity Imbalancer, the Mechanism Design, and the Station Recommender agents. The Charging Station agent starts the protocol and enters the *SendChargingStationRegistration* state. In this state, information is collected on the location, the number, and type of chargers, and other constant characteristics of the station. This information is contained in message *CSinfo* and is published on topic *"CS/+/RegisterChargingStation"*. The service providers that subscribe to that topic receive the message and enter in the *ReceiveChargingStationRequest* state. Then follows the *HandleChargingStationRegistration* state where the transaction takes place and the processing and storage of the information and the registration of the charging station. The data process indicates whether it was successful or failed if there is missing information or if the station is already reg-

istered. Enters the state *SendStationRegistrationOutcome* and sends the *Outcome* on the topic that each service provider has. For example, the Electricity Imbalance send on topic *"EI/+/RegistrationOutcome"* and the Mechanism Design and the Station Recommender replaces the *EI* with *MD* and *SR* respectively. The Charging Station receives the outcome message, enters the *ReceiveRegistrationOutcome* and terminates the protocol by entering the final state.

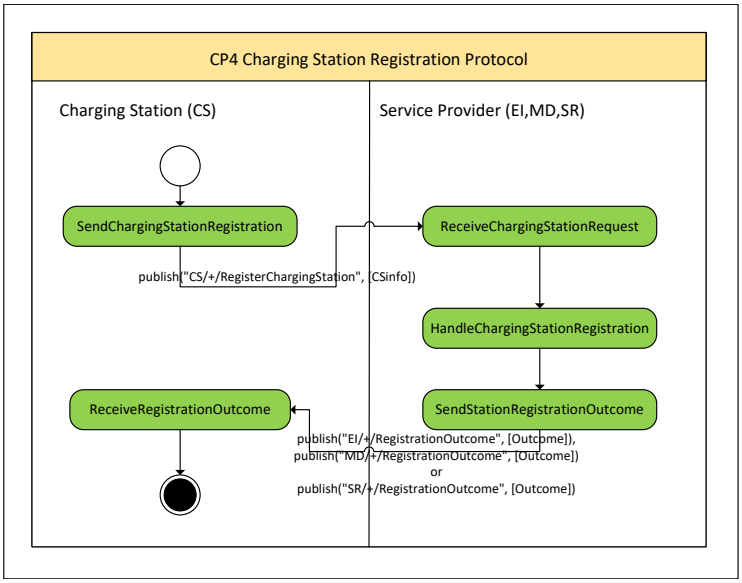

Figure 1.5: The model of the Charging Station Registration Protocol (CP4)

1.1.5 Authenticate Recommendation Protocol (CP5)

As we noted in subsection 1.1.2, the successful completion of a reservation requires authenticating recommendation. The Authenticate Recommendation Protocol (CP5) ensures that the Charging Station agent that the recommendation it received from the Electric Vehicle agent is created from the Station Recommender agent and it not generated or modified from any malicious user.

The Charging Station agent begins the protocol, enters the *SendAuthenticateRecommendationQuery* state and sends the *recommendation* on topic *"CS/+/AuthenticateRecommendation"*, where the plus sign will be replaced with the Charging Station agent's ID. The Station Recommender receive the message, enters the *ReceiveRecommendationAuthenticationQuery* state and checks in the recommendation history data if there is the received recommendation. Then, enter *SendAuthenticationOutcome* and send on *"CS/+/AuthenticateRecommendationOutcome"* the *outcome*, which will be *"True"* if the recommendation found in the history and *"False"* otherwise. The Charging Station re-

ceives the message, enters the *ReceiveAuthenticationResponse* state, and finishes the protocol.

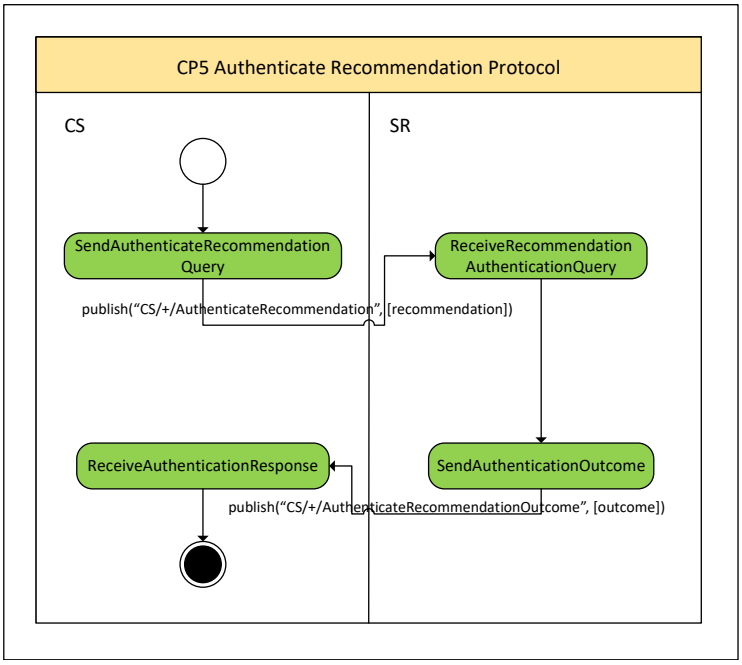

Figure 1.6: The model of the Authenticate Recommendation Protocol (CP5)

1.1.6 Electricity Imbalance Protocol (CP6)

This protocol connects the Mechanism Design, the Station Recommender, and the Charging Station agents, which we will call Service Users, with the Electricity Imbalance agent to be informed about the imbalance of energy (i.e., the shortage or surplus) during each timestep for a day.

The Service User initiates the protocol and enters the *SendElectricityImbalanceRequest* state. It sends on topic *"EI/+/ElectricityImbalanceRequest"* the *day_index* it needs the imbalance. The *day_index* will be "0" for the current day, "1" for the next day "-1" for the previous day etc. The Electricity Imbalance agent receives the message and enters the *ReceiveElectricityImbalanceRequest* state. Follows the *HandleElectricityImbalanceRequest* state, where the calculation of the electric energy imbalance takes place, and enters the *SendElectricityImbalance* state. The *imbalance* is sent on topic *"EI/+/ElectricityImbalance"*, the Service User receives the information, enters the *RecieveElectricityImbalance* state, and finishes the protocol.

There is, also, an other capability that can be enabled to reduce the number of messages and the computational power on the system. The electrical imbalance agent can periodically broadcast the new imbalance or after a big update in the new imbalance. The

imbalance is published on topic *"EI/ElectricityImbalance"* where all the Service Users can be subscribed. In this way, we provide more frequent and useful updates, when it is needed and more used efficient, when there are not changes on imbalance. Although the request and the broadcast features must coexist and the first can still be used from Service Users the did not received the periodic message or just registered on the system and needs to know some historical data.

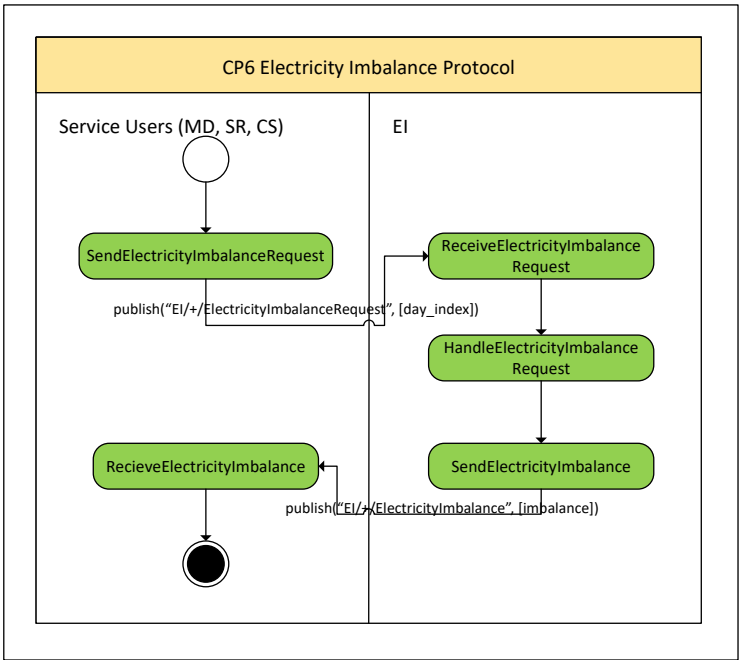

Figure 1.7: The model of the Electricity Imbalance Protocol (CP6)

1.1.7 Electricity Prices Protocol (CP7)

The Electricity Prices Protocol provides the communication between the Charging Station and Mechanism Design agents. The Mechanism Design informs the Charging Station about the purchasing and selling prices each time step for a day.

The Charging Station starts the protocol, enters the *SendElectricityPricesRequest* state and publishes on topic *"MD/+/ElectricityPricesRequest"* the *day_index* related to the current day as described previously on 1.1.6. The Mechanism Design receives it, enters the *ReceiveElectiricityPricesRequest* state, and with knowledge of the imbalance calculates by using the selected pricing algorithm (1.3.1) the prices for each time step in the *HandleElectricityPricesRequest* state. Then, in the *SendElectricityPrices* state, the Mechanism Design publishes the *buy_prices* and *sell_prices* on the *"MD/+/ElectricityPrices"* topic. The Charging Station agent receives the prices, enters the *ReceiveElectricityPrices* and terminates the protocol

The Electricity Prices Protocol can behave as we described the Electricity Imbalance Protocol on 1.1.6. The Mechanism Design has the capability to send periodically or after it receives a big change of the imbalance the updated prices on the topic *"MD/ElectricityPrices"* that can be subscribed from every Charging Station Agent.

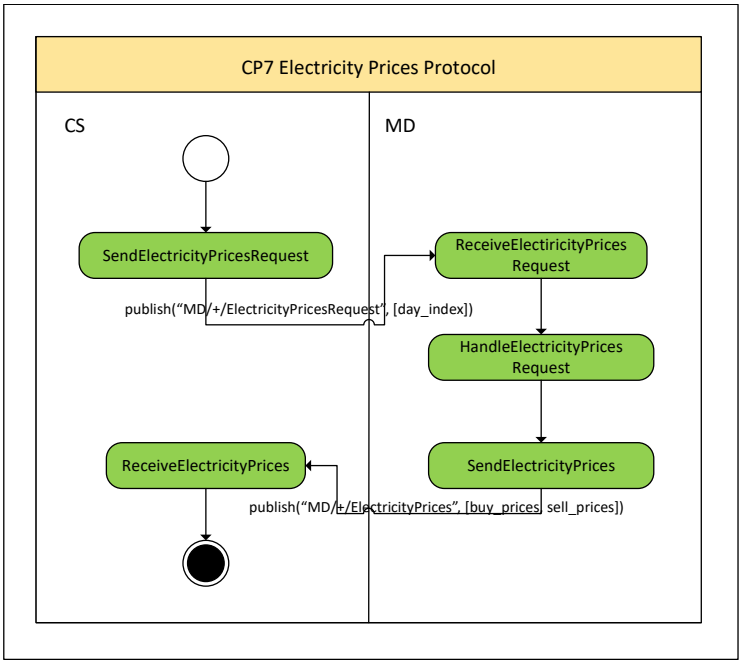

Figure 1.8: The model of the Electricity Prices Protocol (CP7)

1.1.8 Charging Station Update Schedule Protocol (CP8)

The Charging Station Update Schedule Protocol is the protocol that connects the Charging Station agent and the Service Providers that are the Electricity Imbalance and the Mechanism Design agents. The Charging Station agent regularly informs the Service Providers about the amount of energy it needs to get from the Grid and the amount it can provide back each timestep.

The protocol begins with the Charging Station entering the *SendUpdatedChargingSchedule* state and publishing the *schedule* of charging and discharging. Follows the *ReceiveRequestUpdatedStationSchedule* and the *HandleUpdateStationSchedule* where the Service Providers save the updated information. The Electricity Imbalance uses the schedule to calculate the new imbalance and the Mechanism Design calculate the payments must be done and can be needed to some pricing algorithms that the charging stations' production and consumption is required. Then, the Service Provider enters the *SendUpdateScheduleOutcome* state and publishes on the topic *"EI/+/UpdateScheduleOutcome"* for the Electricity Imbalance and *"MD/+/UpdateScheduleOutcome"* for the

Mechanism Design, the *outcome* of the update that can be "Successful update" or "Failed update. Reason" with the reason for example can be that the Charging Station agent is not register in the Service Provider. The Charging Station receives the result, enters the *ReceiveUpdateScheduleOutcome* state, and the protocol is finished.

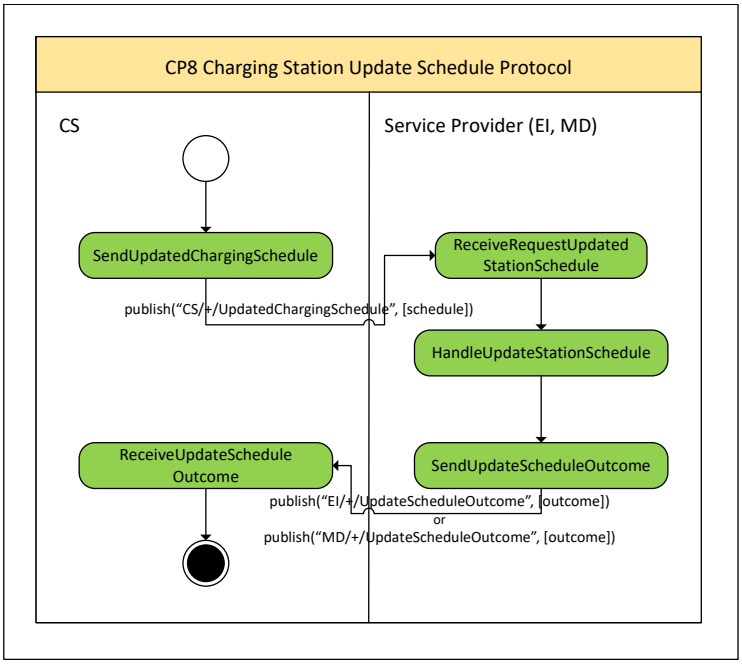

Figure 1.9: The model of the Charging Station Update Schedule Protocol (CP8)

1.1.9 Producer Consumer Registration Protocol (CP9)

This protocol connects the Service Users, the Electric Producer,and the Electric Consumer, with the service providers, Electricity Imbalance and Mechanism Design agents.

The Service User initiate the protocol and enters the *SendRegistrationRequest*. In case of the Service User being producer, it will publish on topic *"EP/+/RegisterElectricityProducer"* the *EP_info* message that contains information about type of the producer (e.g solar panel, wind turbines). Else if it is a consumer, it will send on topic *"EC/+/RegisterElectricityConsumer"* the *EC_info* message that specify the category of consumer it belongs (e.g. residential, commercial, industrial). The Service Provider receives the message, enters the *ReceiveRegistrationRequest* state, and stores the information of the Service User in the *HandleRegistrationRequest* state. Then, it enters the *SendRegistrationOutcome* state publishes the *outcome* of the registration on topic *"EI/+/RegistrationOutcome"* for the Electricity Imbalance agent and on *"MD/+/RegistrationOutcome"* for the Mechanism Design agent. The Service User receives the message, enters the *ReceiveRegistrationOutcome* state, and terminates the protocol.

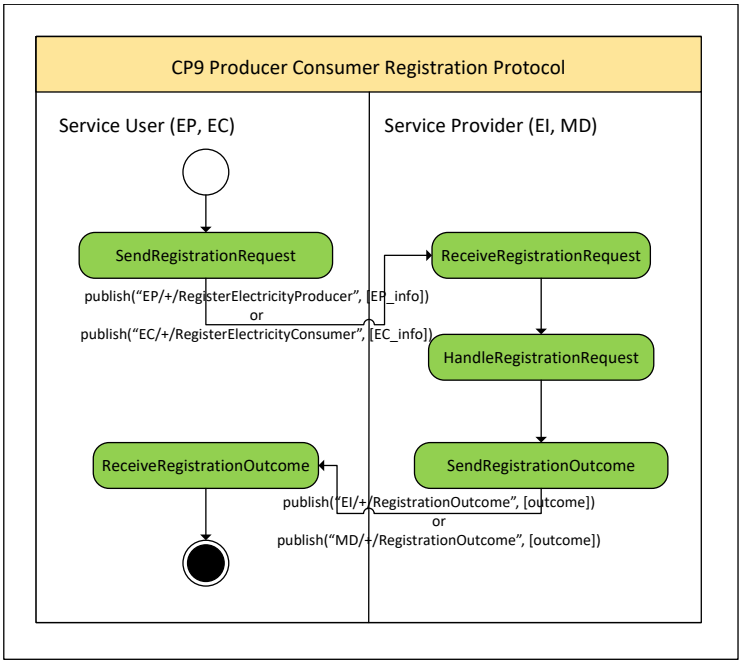

Figure 1.10: The model of the Producer Consumer Registration Protocol (CP9)

1.1.10 Update Expected Production/Consumption Protocol (CP10)

This protocol is responsible for providing connection between Service Users (Electricity Producer, Electricity Consumer agents) and ServiCe Providers (Electricity Imbalance and Mechanism Design agents) to exchange information about their updated expectations of production and consumption. Each time a Service User charges its expected energy profile, informs through this protocol the interested Service Providers.

The Service User starts the protocol, enters the *SendUpdateExpectedEnergyProfile* state, and if it is a producer sends the *expected_production* on topic *"EP/+/UpdateExpectedProduction"* or if it is a consumer sends the *expected_consumption* on topic *"EC/+/UpdateExpectedConsumption"*. The Service Provider receives the message, enters the state *ReceiveUpdateExpectedProfileRequest*. Then, it processes and stores the updated information in the state *HandleUpdateExpectedProfileRequest* and enters the *SendUpdateProfileOutcome* and sends the *outcome* of the update in the topic *"EI/+/UpdateProfileOutcome"* if it is the energy imbalance agent or in *"MD/+/UpdateProfileOutcome"* if it is the Mechanism Design agent. The Service User receives the outcome, enters the *ReceiveUpdateProfileOutcome* state, and finishes the protocol.

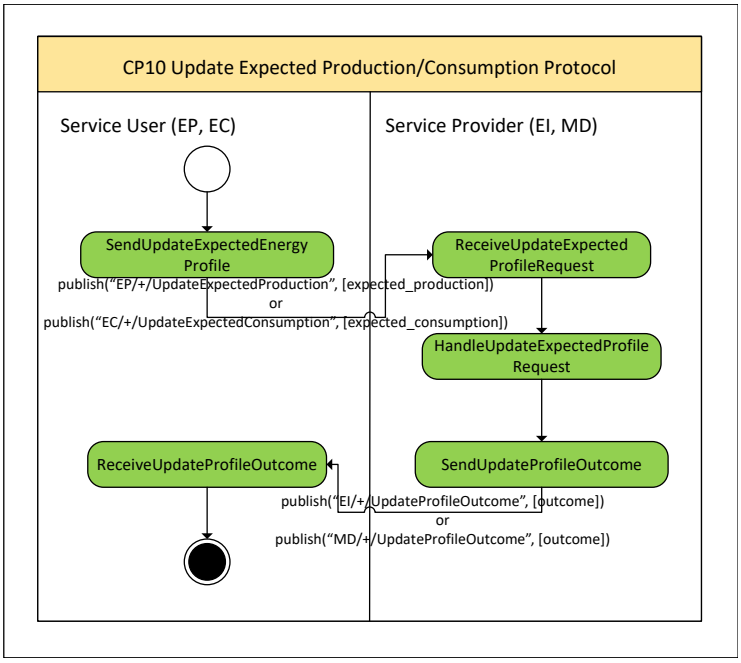

Figure 1.11: The model of the Update Expected Production/Consumption Protocol (CP10)

1.1.11 Update Energy Profile Confidence Protocol (CP11)

In this protocol, service users inform Service Providers about their confidence in the last sent energy profile. The role of Service User can be played from an Electricity Producer, an Electricity Consumer or a Charging Station agent and the role of Service Provider from Electricity Imbalance and Mechanism Design agents.

The Service User starts the protocol and enters the *SendUpdateEnergyProfileConfidence*. The *confidence* sends on topic *"EP/+/UpdateConfidence"*, *"EC/+/UpdateConfidence"*, and *"CS/+/UpdateConfidence"* from Electricity Producer, Electricity Consumer, and Charging Station agents, respectively. The Service Provider receives the message and enters the state *ReceiveUpdateConfidenceRequest*. Manage and store the confidence received in *HandleConfidenceUpdateRequest* and follows the *SendConfidenceUpdateOutcome* state where the Electricity Imbalance and the Mechanism Design publishes *outcome* on topics *"EI/+/UpdateConfidenceOutcome"* and *"MD/+/UpdateConfidenceOutcome"*, respectively. The Service User receives the message, enters *ReceiveUpdateConfidenceOutcome*, and completes the protocol.

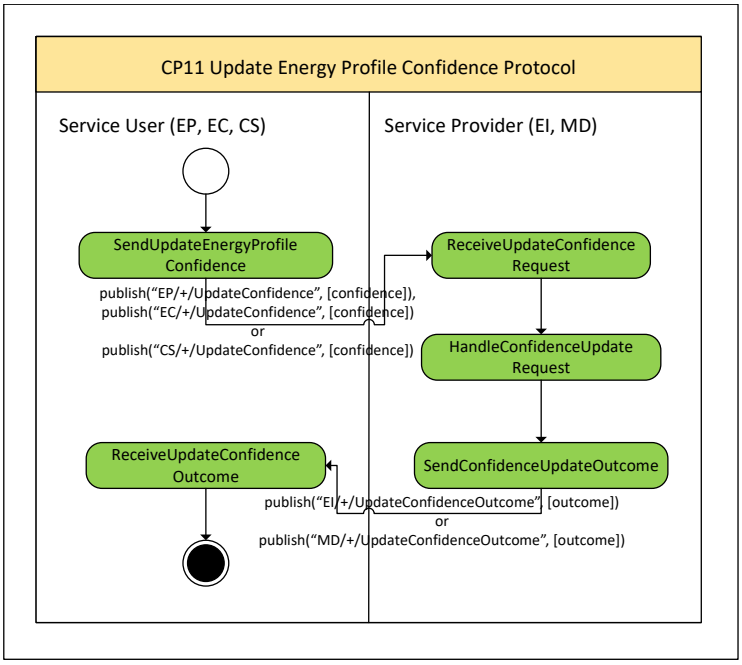

Figure 1.12: The model of the Update Energy Profile Confidence Protocol (CP11)

1.1.12 Update Station Availability Protocol (CP12)

This protocol connects the Charghing Station and the Station Recommender agents. After a change in charging slot availability, for example, a slot is reserved, the Charging Station communicates with the Station Recommender agent to inform it about the new availability schedule. This protocol supports the Station Recommender to recommend an already reserved slot, so we avoid the reservation conflicts.

The Charging Station initiates the protocol, enters the state *SendUpdatedStationAvailability*, and publishes on topic *"CS/+/UpdatedStationAvailability"* the *recommendation* generated by the Station Recommender and contains details of arrival and departure time, and the charging slot is reserved. The Station Recommender receive the update message and enters the *ReceiveRequestUpdateAvailability* state. Follows the *HandleStationAvailabilityUpdate* state where the update on charging slot takes place. Then enters the state *SendUpdateAvailabilityOutcome* and publishes on *"CS/+/UpdateAvailabilityOutcome"* the *outcome* of the update. The Charging Station agent receives the outcome message, enters the *ReceiveUpdateAvailabilityOutcome* state, and terminates the protocol.

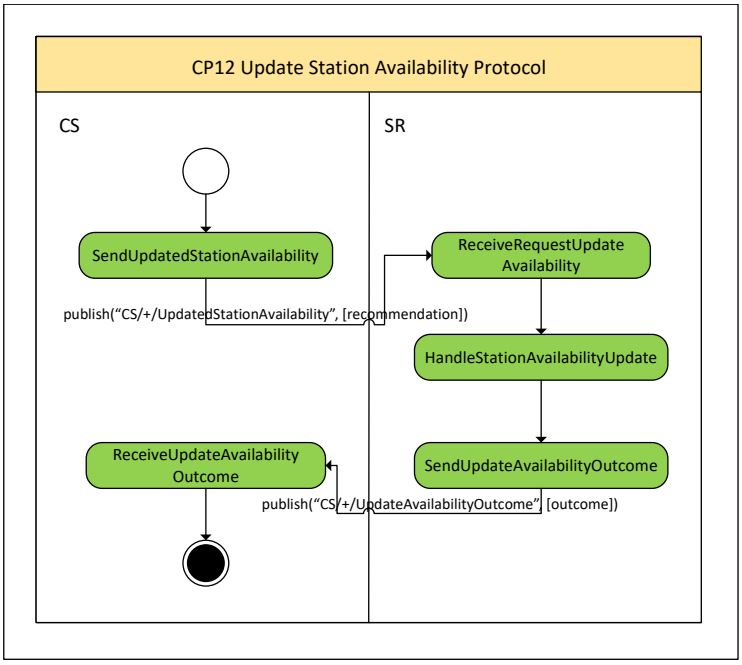

Figure 1.13: The model of the Update Station Availability Protocol (CP12)

1.2 Agent's Behavior

This section describes the internal actions of each stakeholder. There is a state diagram for each agent that shows its operation, behavior and a detailed description of it.

The figures are used to show the flow of action for a agent. START-states show the beginning of the execution (circle). The yellow states show the operations an agent can perform and the blue numbered boxes are for showing the separated tasks an agent can perform inside the *Main* operation. The grey diamonds a decision or branching point. Lines representing different decisions emerge from different points of it. Basic states (shown with green color) are where agent activities are executed and END-states show where execution ends (black dots within a circle).

Transitions from one state to the other occur *(i)* when the activity of the source state finishes and there is no event on the arrow, or *(ii)* when the event on the arrow takes place.

1.2.1 Electric Vehicle Agent

The intra-agent control model for the Electric Vehicle agent is shown in Figure 1.14.

Note that, for simplicity of representation, the protocol roles that the agent realizes are shown as basic states. These can be expanded to the relevant states in the respective protocols. For example, the *CP1 Charging Recommendation Protocol* state must be

replaced by the *EV* state of Figure 1.2.

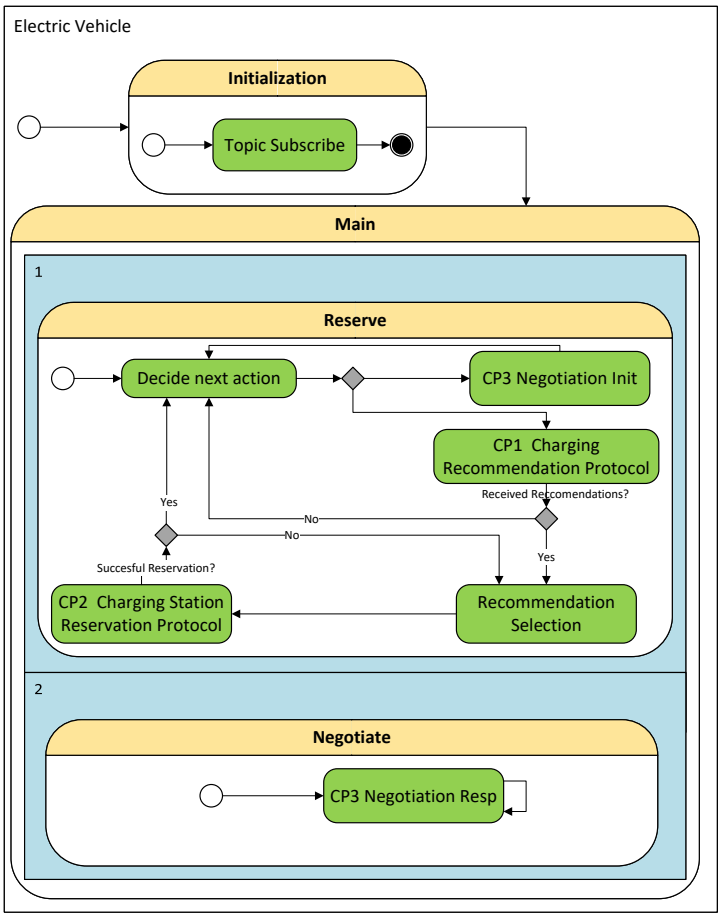

Figure 1.14: The intra-agent model of the Electric Vehicle agent.

At the beginning of its operation, the agent subscribes to the topics that must be connected and performs initialization activities (enters the *Initialization* state) . Then, the agent enters both the *Negotiate* and *Reserve* orthogonal components (Fig.1.14). In the *Reserve* component it makes a transition to the *DecideNextAction* basic state. In this state, the agent makes decisions regarding the charging of the Electric Vehicle (preferences and preferred location).

If the agent suggests making a new recommendation request, it checks if there are recommendations from Station Recommender Agent that meet its needs. If there are not it must calculate again its preferences and send them to Station Recommender until there is a recommendation or it performs another action. Whenever the agent decides to arrange a forthcoming charging, it enters the *CP1 Charging Recommendation Protocol* state, then the *Recommendation Selection* state (to select the best offer), and, finally the *CP2 Charging Station Reservation Protocol* to reserve the selected slot. In case the reservation is not completed, the Electric Vehicle can go back to the *Recommendation*

Selection state and select an other Charging Station. If charging slot is successfully reserved, it returns to the *Decide Next Action* state from which it will have to transit in order to make a new reservation or to negotiate a change in an existing arrangement using the *CP3 Negotiation:Init* state. As the Negotiation protocol (CP3) can be initiated by both parties (Electric Vehehicle or Charging Station) the roles it defines are that of the initiator (*Init*) and of the responder (*Resp*). As the reader can see, the Electric Vehicle can act either as an initiator (entering the *CP3 Negotiation:Init* state) or as responder (entering the *CP3 Negotiation:Resp* state).

1.2.2 Charging Station Agent

The internal behavior of the Charging Station agent is shown in Figure 1.15. Begins with the *Initialization* state, where takes place the *Topic Subscribe* state that the agent connects to the MQTT topics it is interested in and can have access. After it starts the *CP4 Charging Station Registration* to inform the Electricity Imbalancer, the Mechanism Design and the Station Recommender agents about its permanent specification and that it starts to operate. When the protocol finishes, the initialization process ends and follows the operation *Main*, which contains four separate functions that are numbered and displayed in light blue boxes.

The *Charging Reservation* process starts with the *ReceiveChargingReservationRequest* and enters *HandleChargingReservationRequest* state. The Charging Station activates the *CP5 Authenticate Recommendation Protocol*. If authentication fails, the protocol ends; otherwise, if it authenticates successfully, enter *ScheduleAndMakeReservation* where it performs the schedule according to the selected scheduling algorithm and reserves the Charging Slot. If charging scheduling and reservation are completed successfully, the *CP8 Charging Station Update Schedule Protocol* starts to inform the Electricity Imbalancer, the Mechanism Design agents about the energy consumption per time step. Then, it starts *CP12 Update Station Availability Protocol* to inform the station recommender agent about the availability of the new charging slot and finishes the state *HandleChargingReservationRequest*. Enters the *SendReservationOutcome* state where the schedule and the reservation will be sent to the interested Electric Vehicle otherwise sends the reason of failure and terminates the process.

The second displayed (we remind the reader that all processes can be operated separately and parallel) process starts when the Charging Station agent needs to change a

reservation. It initiates *CP3 Negotiation Protocol* and after its completion makes two checks. If there is a change on the charging schedule, then it starts the *CP8 Charging Station Update Schedule Protocol* to inform the Mechanism Design and the Electricity Imbalance. If there is a change on the arrival or departure time the Station Recommender must be informed by *CP12 Update Station Availability Protocol* and finishes the process.

The third and fourth processes start when the Charging Station receives electricity imbalance and prices, from the Electricity Imbalancer and the Mechanism Design agents, and uses the *CP6 Electricity Imbalance Protocol* and *CP7 Electricity Prices Protocol* to store the newly received data.

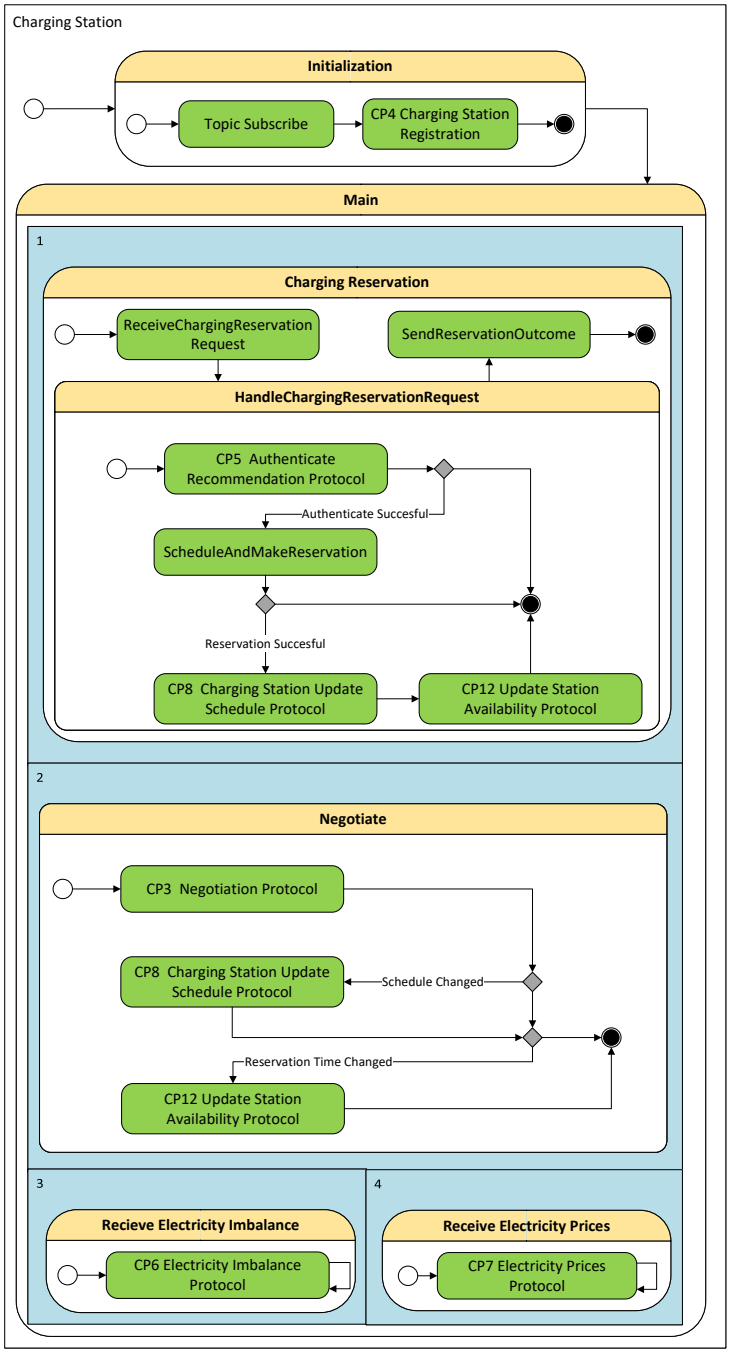

Figure 1.15: The intra-agent model of the Charging Station.

1.2.3 Station Recommender Agent

The intra-agent functionality of the Station Recommender agent is shown in Figure 1.16. In the *Initialization* state the *Topic Subscribe* process takes place to make sure that the agent will receive the messages that needs to be informed.

Then it enters the *Main* state where the protocols will be enabled on the arrival of the corresponding message. If the Station Recommender receives a recommendation request,

it will take into account the preferences of the Electric Vehicle. It will use *CP1 Charging Recommendation Protocol* to send the most suitable Charging Stations for Electric Vehicles that match their preferences. The Station Recommender receives information about the new newly opened Charging Station and uses *CP4 Charging Station Registration Protocol* to record and possibly add them to future recommendations. It, also, provides authentication services for the recommendations that has sent. When a recommendation needs to be verified that it is genuine the *CP5 Authenticate Recommendation Protocol* activates. The Station Recommender must be informed about the availability of the Charging Stations to generate correct recommendations and do not recommend charging slots that are already reserved in that specific time period. So on the arrival of a availability update message the *CP12 Update Station Availability Protocol* activates and stores the update.

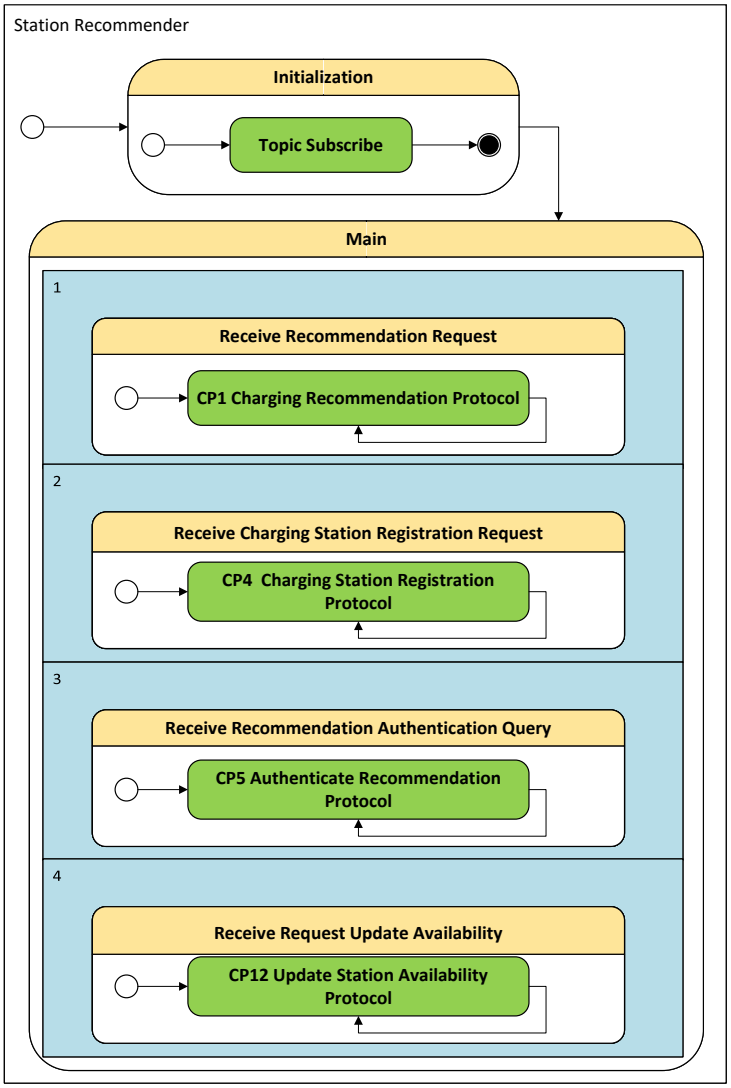

Figure 1.16: The intra-agent model of the Station Recommender.

1.2.4 Electricity Imbalance Agent

The Electricity Imbalance agent behaves according to Figure 1.17. It starts with *Initialization* where the agent subscribes to the MQTT topic that must receive the published messages on them.

On the first two operations we see the registration request of the Charging Station, the Electricity Producer and Consumer that activates the *CP4 Charging Station Registration Protocol* and the *CP9 Producer Consumer Registration Protocol*. The registration process is necessary in our system to make sure we have knowledge of the maximum production, consumption, and details, attributes about the prosumers. The next two operations describe the update of the energy consumption or generation for the Charging Station in *CP8 Charging Station Update Schedule Protocol* and for Electricity Producers and Consumers it uses the *CP10 Update Expected Production/ Consumption Protocol*

The fifth displayed operation informs the Electricity Imbalance agent about the confidence of the Charging Station, the Electricity Producers and Consumers on the amount of energy they are going to generate or consume. The last operation is the most significant for the agent and it is used to inform the interested agents about the total generation and consumption of the energy in the smart grid. After it receives an electricity imbalance request, it activates the *CP6 Electricity Imbalance Protocol* where the calculation of the imbalance and the publication of the requested message takes place.

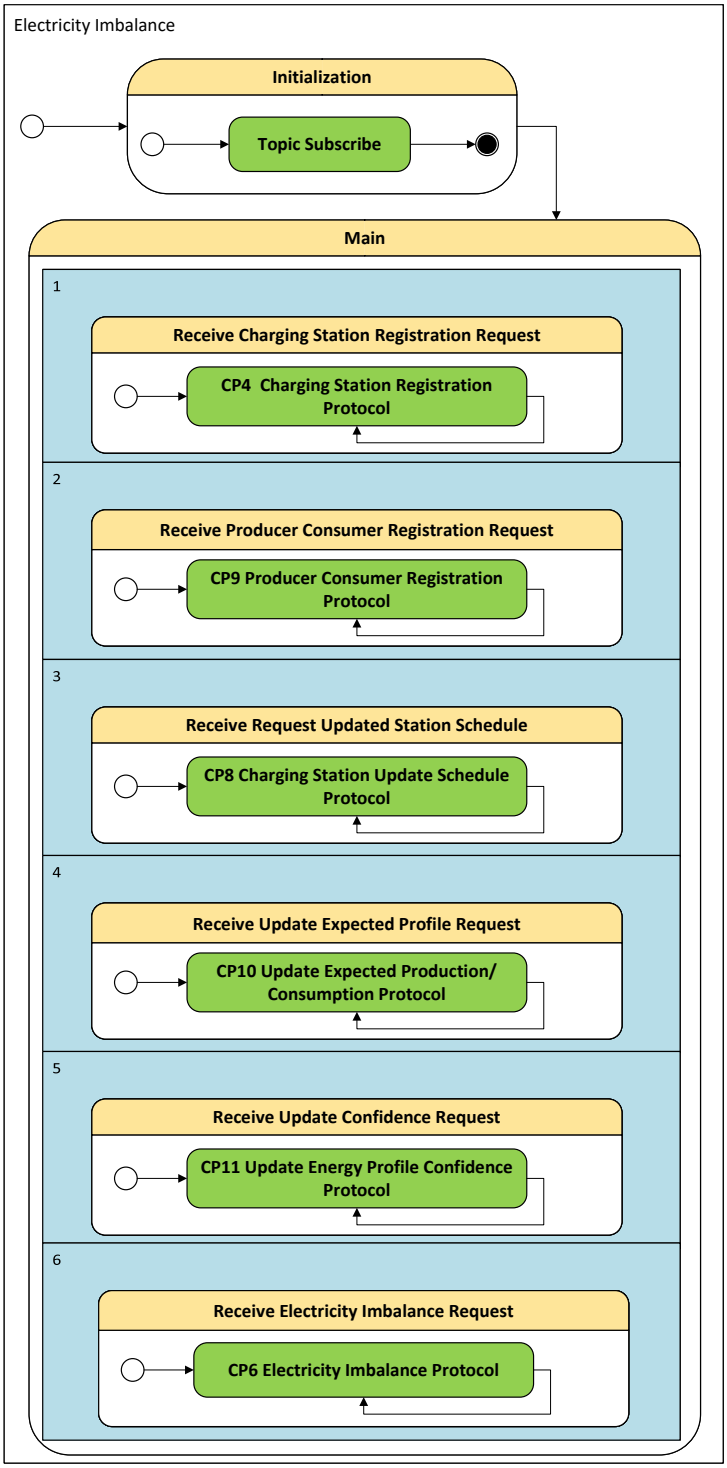

Figure 1.17: The intra-agent model of the Electricity Imbalance.

1.2.5 Mechanism Design Agent

Figure 1.18 displays the intra agent behavior of the Mechanism Design agent. It begins with the *Initialization* process that contains the subscription on MQTT topics it needs to operate. Then, it enters the *Main* state where the agent responds to the incoming messages. It receives registration request from Charging Stations, Electricity Producers,

and Consumers, enables the *CP4 Charging Station Registration Protocol* and the *CP9 Producer Consumer Registration Protocol* to store the necessary information about their capabilities. The *CP8 Charging Station Update Schedule Protocol* and the *CP10 Update Expected Production/ Consumption Protocol* are used to communicate with the Charging Stations, Electricity Producers and Consumers about their expected production and future energy needs. Also, the three agents mentioned send how confident they are about their prediction and the Mechanism Design uses the *CP11 Update Energy Profile Confidence Protocol*. When the agent receives an Electricity Imbalance message, it activates the *CP6 Electricity Imbalance Protocol*. If the new imbalance changes significantly and affects the prices, then it activates *CP7 Electricity Prices Protocol* to inform the interested agents about the new prices. The *CP7 Electricity Prices Protocol* is used if an electricity price request is received to send the most recent calculated buying and selling prices for an energy unit (usually 1 kWh).

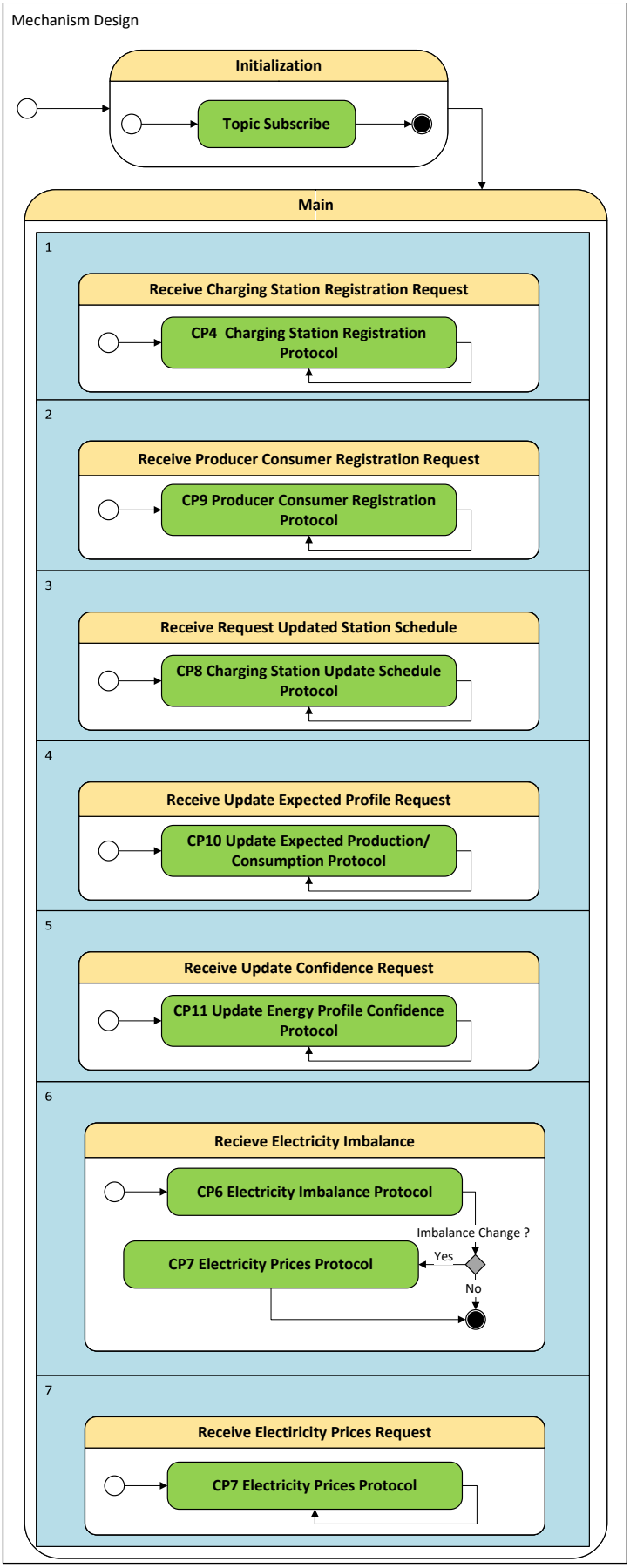

Figure 1.18: The intra-agent model of the Mechanism Design.

1.2.6 Electricity Producer Agent

Figure 1.19 describes the internal agent model of the implemented Electricity Producer. It starts with the *Initialization* where take place the *Topic Subscription* and the *CP9 Producer Consumer Registration Protocol*. The first process is needed to make sure the agent can receive the incoming messages, and the protocol informs the interested service providers about the type (e.g. photovoltaic panels, wind turbines), the rated power, and the location. The *Main* process follows where contains the *Periodic Energy Update* that activates periodically (for example, every day). It begins with *Calculate Production* where the agent can make an estimate of future production taking into account date, time, weather forecast and previously recorded and historical data and uses *CP10 Update Expected Production/Consumption Protocol* to publish it. After, it compares the expected production amount with similar past dates. If those values are close, then we have a good estimate of the expected production. The Electricity Producer agent uses the *CP11 Update Energy Profile Confidence Protocol* to send the confidence to the Mechanism Design and the Electricity Imbalance agent.

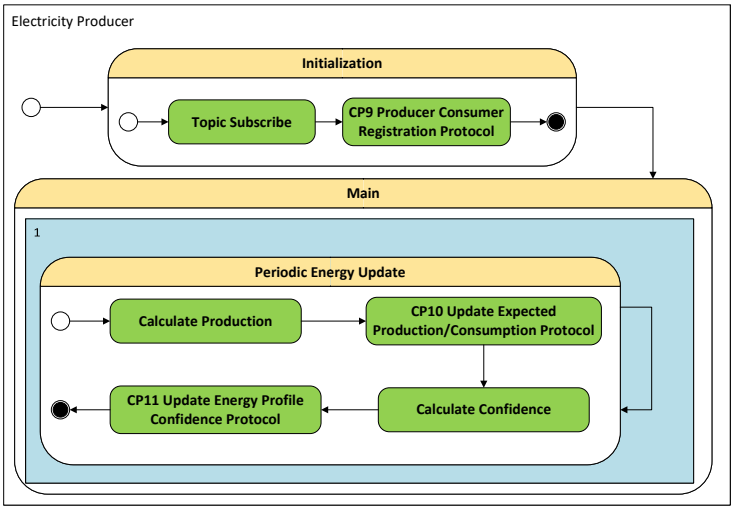

Figure 1.19: The intra-agent model of the Electricity Producer.

1.2.7 Electricity Consumer Agent

The internal behavior of the Electricity Consumer agent is shown in Figure 1.15. The agent starts with *Initialization* where the subscription is made on MQTT Topics and *CP9 Producer Consumer Registration Protocol*. CP9 is used to inform the Mechanism Design and Electricity Imbalance agents about the type of the consumer (residential, commercial,

industrial). Then, the *Main* process follows that contains the *Periodic Energy Update* that is a repeated process and activates after a specific time period, for example every day. It starts with the *Calculate Consumption* which takes into account the consumption of previous days and from previous years if this information is available. It publishes the consumption using the *CP10 Update Expected Production/Consumption Protocol*. After, the Electricity Consumer agent calculates the confidence from previous collected data and publish it with the use of *CP11 Update Energy Profile Confidence Protocol*.

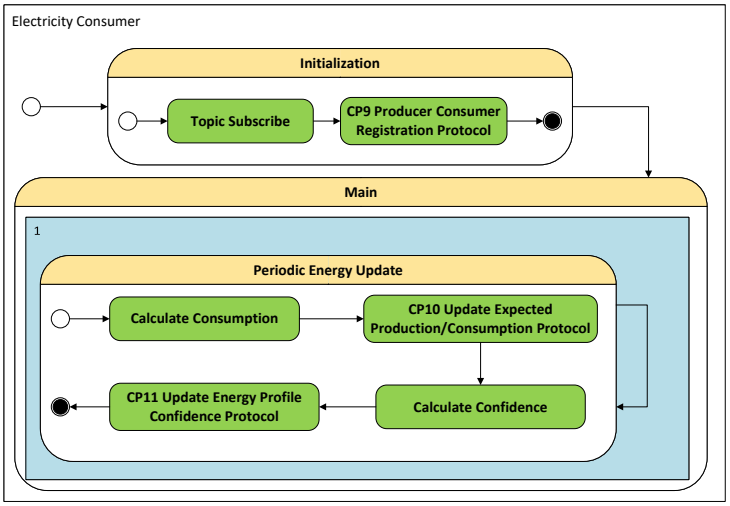

Figure 1.20: The intra-agent model of the Electricity Consumer.

1.3 Implemented Agent Strategies

For the purposes of evaluation via simulation, we need to test different methods and compare their effects on the system in simulation mode. To this end, we implemented two pricing algorithms used by the Mechanism Design agent to observe how they contribute to grid stability, i.e. to reducing the energy surplus and deficit peaks. We also implemented three scheduling approaches that determine when and how much energy is exchanged between Charging Station and Electric Vehicle agents.

1.3.1 Price Calculation Algorithms for the Mechanism Design Agent

A) *NRGCoin pricing algorithm:* This pricing mechanism is inspired by the one in [0], and aims at incentivizing producers and consumers to balance supply and demand. Let the aggregate supply and demand each time interval t be S_t, and D_t and the individual agent i's desirable amounts of energy for selling and buying be s_t^i and d_t^i. The closer D_t

and S_t are, the better prices are offered to buy and sell. The price for selling energy is:

$$P_t^{sell}(s_t^i, S_t, D_t) = (0.1 \cdot s_t^i) + \frac{0.2 \cdot s_t^i}{e^{(\frac{S_t - D_t}{D_t})^2}}$$

and the price for buying energy is:

$$P_t^{buy}(d_t^i, S_t, D_t) = \frac{(0.65 \cdot D_t) \cdot d_t^i}{D_t + S_t}$$

B) *Adaptive pricing algorithm:* According to this mechanism proposed in [0], we estimate the evaluation of energy with respect to the cost of the Electric Vehicle agents, by calculating an $\hat{\alpha}$ value:

$$\hat{\alpha} = \frac{\sum_{t=2}^{N} \frac{P_1^{buy} - P_t^{buy}}{2 \cdot (d_1^{i'} - d_t^{i'})}}{N-1}$$

where N is the number of time intervals in the planning horizon, and $d_t^{i'}$, the demand of Electric Vehicle agent i during the interval t. The mechanism can adjust prices to motivate agents to charge their electric vehicles when there is an energy surplus on the grid. The buying prices for the intervals $t \in \{1, ..., T\}$ are given by:

$$\hat{P}_1^{buy} - 2 \cdot \hat{\alpha} \cdot (S_1 - D_1) = ... = \hat{P}_T^{buy} - 2 \cdot \hat{\alpha} \cdot (S_T - D_T)$$

1.3.2 Charging Scheduling Algorithms for the Electric Vehicle Agent

A) *First slot:* In this case, EVs charge their battery in the first available time interval, without taking into account if prices are better or worse.

B) *Lowest Prices:* In this approach, electric vehicles are trying to reduce their charging costs by choosing to charge during time periods when the energy prices are the lowest possible.

C) *V2G:* In this case, electric vehicles can discharge their batteries when prices are high to provide load to the rest of the grid, and then charge them back when prices are lower, nevertheless within the periods that electric vehicles are connected to a charger. For this purpose and inspired by [0] and [0], we used linear programming to minimize an objective function representing charging costs in the presence of constraints regarding the electric vehicle owner preferences and Electric Vehicle's charging specifications. The

optimization function is the following:

$$\min \sum_{t}^{T} C_t^{G2V} + C_t^{deg} - I_t^{V2G} \tag{1.1}$$

subject to:

$$C_t^{G2V} = d_t^{G2V} * P_{max}^{G2V} * p_t^{buy} * dt \tag{1.2}$$

$$C_t^{deg} = f^{deg} * d_t^{V2G} * P_{max}^{V2G} * dt \tag{1.3}$$

$$I_t^{V2G} = d_t^{V2G} * P_{max}^{V2G} * p_t^{sell} * dt \tag{1.4}$$

$$d_t^{G2V} + d_t^{V2G} \leq 1, \;\; d_t^{G2V}, d_t^{V2G} \in [0,1] \tag{1.5}$$

$$\sum_{t}^{T} d_t^{G2V} * P_{max}^{G2V} * dt - d_t^{V2G} * P_{max}^{V2G} * dt = E_{need} \tag{1.6}$$

$$\sum_{t}^{k} d_t^{G2V} * P_{max}^{G2V} * dt - d_t^{V2G} * P_{max}^{V2G} * dt + E_{init} \leq c_{max}, k \in [1,T] \tag{1.7}$$

$$\sum_{t}^{k} (d_t^{G2V} * P_{max}^{G2V} * dt - d_t^{V2G} * P_{max}^{V2G} * dt) + E_{init} \geq c_{min}, k \in [1,T] \tag{1.8}$$

where t is the charging interval of the charging period, C_t^{G2V} is the cost of charging, C_t^{deg} is the battery degradation cost, and I_t^{V2G} is the profit earned by selling energy to the grid. The d_t^{G2V} and d_t^{V2G} are decision variables for G2V and V2G in our optimization problem and can take values between 0 and 1, where intermediate values are assigned when it is optimal to charge or discharge at a fraction of the maximum charging (P_{max}^{G2V}) or discharging (P_{max}^{V2G}) power; p_t^{buy} and p_t^{sell} are the buying and selling prices of energy. f^{deg} is the degradation factor based on the method presented in [0], which is used to evaluate the degradation cost C_t^{deg} and dt is the duration of each time interval.

The constraints in expressions (1.5) - (1.8) must be satisfied during Electric Vehicle scheduling optimization. In (1.5) it is guaranteed that an electric vehicle will charge, discharge or stay idle in each time interval. The constraint (1.6) states that at the end of the charging session the electric vehicle's battery must be charged at the desired capacity E_{need} that the owner has set as the target. In constraints (1.7) and (1.8) we limit the allowable range of battery charging state to be between the minimum (c_{min}) and the maximum (c_{max}) capacity by adding the net energy that has been received until the end of each time interval, plus the initial amount of energy already stored in the battery.