***PowerTAC**: The Power Trading Agent Competition*
Term Project for COMP 512: Multiagent Systems

!!! Deadline: 10 January 2022, 23:55 !!!

Create teams with up to 3 members!
For questions contact: sorfanoudakis@tuc.gr

1/11/2022

1 Introduction

PowerTAC is a rich competitive economic simulation of future energy markets featuring several Smart Grid components. With the help of this simulator, researchers can better understand the behavior of future customer models and experiment with retail and wholesale market decision-making by creating competitive agents and benchmarking their unique strategies against each other. In this way, a host of useful information is extracted, which can be used by policymakers and industries to prepare for the upcoming market changes.

This year, the students of the "Multiagent Systems" course are called to create their own PowerTAC agents, enhancing them with Artificial Intelligence and Multiagent system techniques combined with plenty of imagination to participate in the local TUC-PoweTAC tournament at the end of this semester. In particular, we provide you with a sample broker so you can start implementing your own agents to enter this multiagent competitive environment right away!

2 PowerTAC: Rules & Gameplay

In this section, we will demonstrate the basic components of PowerTAC and its main rules. The PowerTAC vision consists of competitive agents that will harness the energy supply and demand of the simulation environment to make a profit. Specifically, the brokers-agents will buy and sell energy through consumption tariffs with individual retail customers like households, retail stores, or even bigger enterprises, as well as electric vehicles. At the same time, the agents will interact and trade within the wholesale market, a real-world replica of the European and North American wholesale energy markets. The main elements of PowerTAC can be seen in Figure 1.

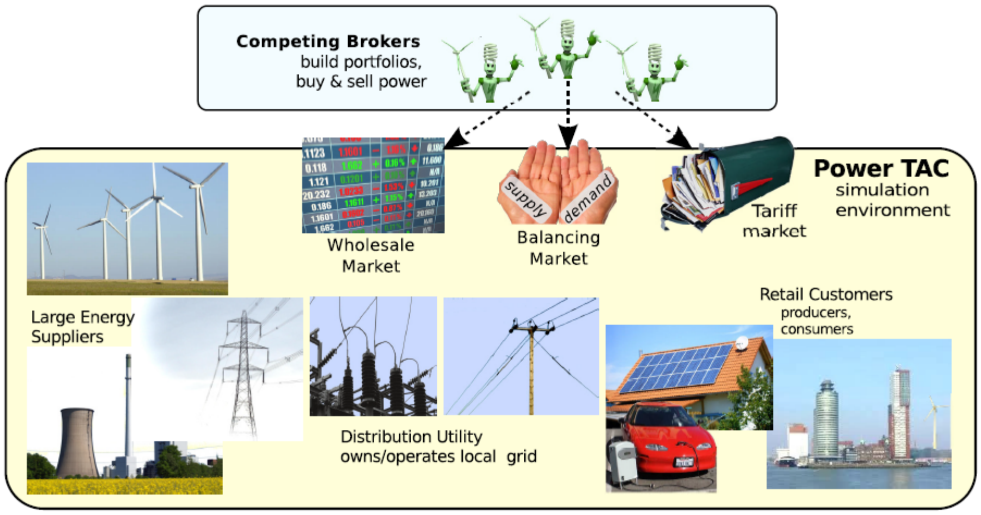

Figure 1: Main PowerTAC entities

2.1 Timeslots

The simulations are divided into discrete steps called *timeslots*. A "timeslot" represents an hour of simulated time, with each one of them taking **5 seconds** in real-time. Each game consists of at least 1440 timeslots (two months of simulated time and at least 2 hours of real-time per game). This means that in each PowerTAC game, at any time, there is an active timeslot and a set of future timeslots for which the brokers can reserve and trade energy. The main objective of the brokers is to try to balance the electricity demand and supply in each of the future timeslots to get the highest profits while minimizing monetary penalties.

2.2 Brokers

Brokers(agents) are the real-life analogy to energy retailers, thus they can perform similar actions. In each timeslot, every agent can decide and perform any of the actions seen in Figure 2.

Specifically, for the needs of your term project, the actions are split into two categories (Retail market actions and Wholesale market actions):

- **Retail Market actions:**
  - **Publish** new tariffs
  - **Revoke** an active tariff
  - **Modify existing tariff terms** by revoking the old tariff and publishing a new one in its place.
- **Trade in the Wholesale Market:**
  - **Place bids** to buy energy in future timeslots

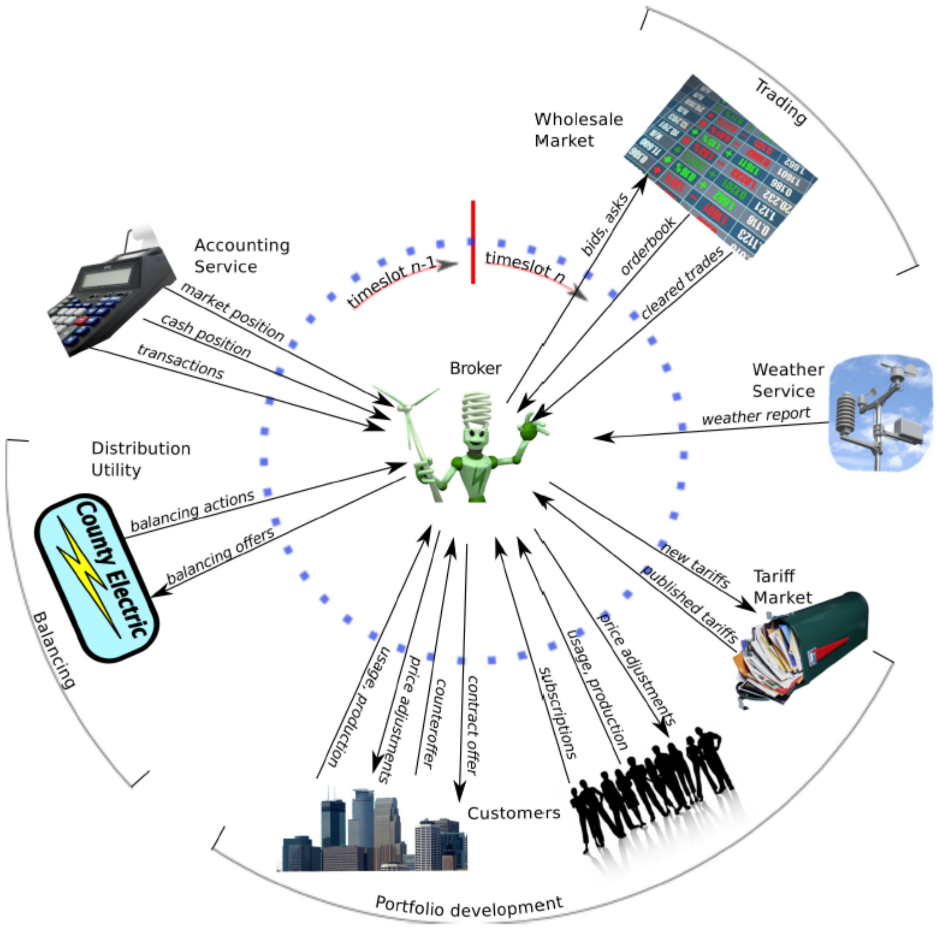

Figure 2: Available actions of a broker during a timeslot

– **Place asks** to sell surplus energy in future timeslots

Furthermore, there is a lot of information available to the brokers in order to help them make the correct decisions. In a few words, agents know about:

- **Other Broker identities**, only the names of the competitor agents
- **Weather reports/forecasts**.
- **Active Tariffs and the specification of each broker**.
- **Wholesale market clearing data**.
- **Wholesale market order-books**.
- **Energy consumption/production of their customers**.
- **Every Transaction is private data for each agent**, like cash position, market positions, portfolio supply demand, balancing, and distribution transactions.

All this information can be used to develop your own strategies and also create opponent models.

2.3 Retail Market Tariffs

Tariffs are the main and easier way for an agent to make a profit. In PowerTAC, there are specific tariffs for a wide range of different power consumers/producers types. These are general consumption, interruptible consumption, general storage, battery storage, thermal storage consumption, electric vehicles, general production, wind production, and solar production.

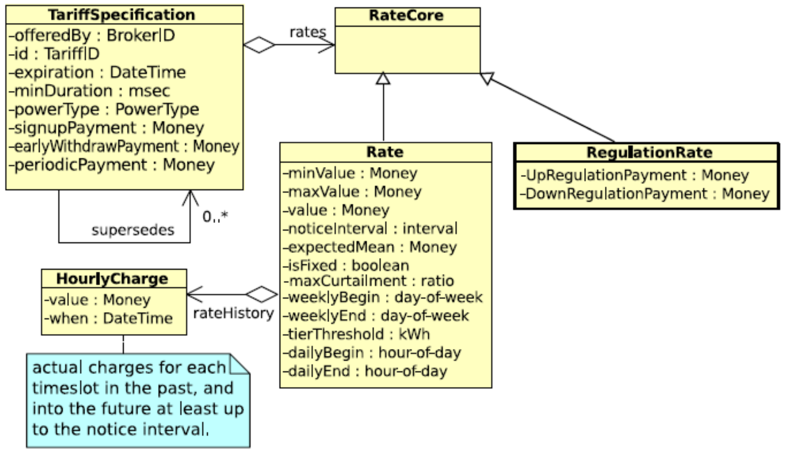

Figure 3: Detailed Tariff Parameters

It is easily visible that with these tariff terms countless different tariffs can be offered. Some of the common tariff features that are supported are:

- **Tiered Rates**, which means that customers can be charged with more than one rate for electricity: a lower rate for the electricity used up to a certain threshold; and a second, higher rate for all additional use.
- **Time of Use rates** is a method of measuring and charging a customer's energy consumption based on when the energy is used.
- **Two-part tariffs** are a combination of the fixed daily fee plus the usage fee.
- **Sign up and Early withdrawal penalties**, these features can be used as a tariff clause.
- **Regulation Rates** is used by storage devices and can specify different prices for the use of the device for up-regulation or down-regulation.
- **Dynamic Pricing tariffs** in which the new prices must be communicated to subscribed customers some number of timeslots before the timeslot to which they apply.

!!! Hint: We advise you to focus only on your agents' most impacting tariff parameters and leave the rest parameters to the default values. The important parameters are the following:

- **offeredBy**: The Broker (agent) that offers this tariff
- **id**: a unique tariff identifier
- **powerType**: The power type of the customers a tariff refers to
- **signupPayment**: The payment when a customer signs up for a tariff
- **earlyWithdrawalPenalty** and **minDuration**: The penalty when a customer withdraws from an active tariff and signs up for another one before the minimum duration has passed
- **periodicPayment**: The fixed payment for a tariff
- **Rates** [1]:
  - **weeklyBegin, weeklyEnd, dailyBegin, dailyEnd**: These parameters are used to define a specific rate tariff's start and ending date
  - **value**: the amount of money paid per KWh for the specified period of time

2.4 Power Types

You can offer tariffs for a variety of power-type customers, however, for this project, we suggest you focus on offering tariffs with the following power types:

- **PRODUCTION**: For solar panels and wind turbines
- **CONSUMPTION**: For general consumption customers like households, shops, etc.
- **STORAGE**: For assets with storage capacities like batteries and Electric Vehicles(EVs)

You can have more than one active tariff per power type with different rates each.

2.5 Weather Reports

Another significant feature of PowerTAC is the weather reports. In each timeslot, a weather report for the current timeslot is sent to the brokers, along with a weather forecast about the next 24 timeslots. This information can be used by the agents for prediction or reinforcement learning tasks since the customer models are directly affected by the weather.

2.6 Wholesale Market

The wholesale market is the place that allows brokers to buy and sell energy for the **next 24 timeslots**. Specifically, the PowerTAC wholesale market is a "day ahead" **periodic double auction**, similar to the ones you will see at the MAS course, which clears in each timeslot.

[1] You can use simple fixed-rate tariffs by defining only a single "Rate", you can also add multiple rates for different times of day, for example, electricity can be cheaper at night, etc.

2.7 Balancing Market and Balancing Fees

The balancing market is the real-life equivalent of an Independent System Operator (ISO) in the U.S. or a Transmission Systems Operator (TSO) in Europe. In PowerTAC the balancing Market monitors the electricity grid and maintains balance by keeping the supply and demand at similar levels in each timeslot by exercising capacity controls on behalf of the agents. In this way, when a broker fails to procure the required energy in time, the balancing utility charges the broker for the missing energy at higher rates, acting as a penalty. That fee is called *Balancing Fee* and can vary from smaller charges to very high destructive penalties depending on the Wholesale market prices.

!!! In a few words, Balancing Fees are the first of the two primary penalty types that you will need to address. In particular, **Balancing fees** will be imposed on your agents when you have subscribed retail customers, but you could not buy the required energy for that timeslot from the wholesale market.

2.8 Distribution utility and Transmission Capacity Fees

The Distribution utility (DU) is responsible for three different operations. The first one, as its name suggests, is to distribute the energy to each customer while it charges each broker the distribution fees which are relevant to the energy transmitted through the grid.

The second functionality of the Distribution utility is to publish default tariffs so the customers can have an active tariff to subscribe to when there are no published tariffs by any other broker.

!!! Finally, DU is responsible for issuing the **Transmission Capacity Fees** (TSF). These fees represent the amount of money a broker should pay for its customers' contribution to demand peaks. TSF charges the *three* highest demand peaks at the end of a 168 timeslot period (1 week of simulated time). If you notice the simulation graphs, you will see a significant drop in the total money of the brokers happening every 168 timeslots.

In the current PowerTAC competition, these fees are the main problem an agent faces when it tries to dominate the retail market.

2.9 PowerTAC - Game Specification

For even more detailed information about the game specification, you can read the following rule book (link). You are **encouraged** to read the sections that concern you, however, you are not required to fully comprehend every aspect of it to get the maximum grade since, in order to reduce the size of this project, we have narrowed down the actions to only the most important ones.

3 Agent Implementation

In your project, you are called to write code for the **retail** and the **wholesale** market decision-making modules of your PowerTAC agent. There are only two files that you need to work on, these are the **PortfolioManagerService.java** and the **MarketManagerService.java**, however, you can create and use as many new files as you need. Specifically, there are already many server-message-handling functions implemented that will help you focus only on the development of your own strategies. In the following section, we will try to provide you with concise information regarding the most significant parts of the code that you will need for this project, however, feel free to check the official project documentation (link).

The provided sample-broker contains example code showing you how to perform the basic actions, which are publishing and revoking tariffs (in the “PotfolioManager.Service.java”) while submitting bids and asks in the wholesale market (in the “MarketManagerService.java”).

3.1 activate()

Both files have an *activate()* function. These functions are called automatically **every timeslot** after all the server-message handling functions have completed their execution. This is where most of your code will be written.

3.2 initialize()

At the beginning of every simulation, the initialize() functions for both files are called. You might want to initialize some variables or array lists there.

3.3 PortfolioManagerService.java Functions

In this java file, you will write code for publishing and revoking competitive tariffs. You will also monitor the tariffs offered by the other broker-agents and devise your own strategy.

3.3.1 handleMessage(TariffSpecification spec)

These are sent by the server when new tariffs are published. If it’s not ours, then it’s a competitor’s tariff. We keep track of competing tariffs locally, and we also store them in the “tariffRepo”.

3.3.2 handleMessage(TariffTransaction ttx)

This function has implemented code so you can store information regarding the tariff transactions of your agent. We only care about certain types: PRODUCE, CONSUME, SIGNUP, and WITHDRAW.

3.4 MarketManagerService.java Functions

In this file, you will write code for submitting bids and asks in the wholesale market of PowerTAC.

3.4.1 handleMessage (WeatherForecast forecast)

In each timeslot, it receives a new WeatherForecast for the next 24 timeslots.

3.4.2 handleMessage (WeatherReport report)

Receives the weather report (the actual weather) for the current timeslot.

3.4.3 handleMessage (CapacityTransaction dt)

This function handles the capacity transaction fees, which are a penalty for contributing to the overall peak demand over the recent past.

3.4.4 handleMessage (ClearedTrade ct)

It handles a ClearedTrade message and is where you can keep track of the clearing market prices.

4 Requirements for the project & Bonus

1. (65%) **Developing an intelligent PowerTAC agent** that can dynamically publish competitive tariffs (45%) and participate in the wholesale market to buy the required energy (20%).
2. (35%) **Writing a detailed 8-page report** containing information about the algorithms and the decisions you made. Also, you should **evaluate** your agent's strategy (profits, penalties, etc ..) when competing with other PowerTAC agents that we will provide you with, and **present** the most significant results in the form of tables.
3. **Bonus(up to 10%):** based on the final rankings of the TUC-PowerTAC tournament.
4. **Bonus(up to 15%):** If in order to publish your dynamic tariffs, you exploit specific opponents' models that you create via a dedicated **opponent modeling approach/module**.

5 Useful Links

You are free to decide which algorithms to use for each part of your agents. There has been significant research in the past about PowerTAC strategies, so you can read some of the following papers to get inspiration:

1. Orfanoudakis, S. Diploma Thesis 2021 (link)
2. Orfanoudakis, S., Kontos, S., Akasiadis, C., & Chalkiadakis, G. (2021). Aiming for Half Gets You to the Top: Winning PowerTAC 2020. EUMAS.(link)
3. Chandlekar, S., Pedasingu, B.S., Subramanian, E., Bhat, S., Paruchuri, P., & Gujar, S. (2022). VidyutVanika21: An Autonomous Intelligent Broker for Smart-grids. IJCAI.(link)
4. Chowdhury, M.M., Kiekintveld, C., Tran, S., & Yeoh, W.G. (2018). Bidding in Periodic Double Auctions Using Heuristics and Dynamic Monte Carlo Tree Search. IJCAI. (link)
5. Özdemir, S., & Unland, R. (2017). The strategy and architecture of a winner broker in a renowned agent-based smart grid competition. Web Intell., 15, 165-183. (link)

6. Kuate, R.T., He, M., Chli, M., & Wang, H.H. (2013). An Intelligent Broker Agent for Energy Trading: An MDP Approach. IJCAI. (link)

7. Makrodimitris L., Symeonidis A., Design and Development of an Agent Broker for the PowerTAC Tournament, 2020, Diploma Thesis, Aristotle University of Thessaloniki (link)

- For the git repository of the sample broker: https://github.com/powertac/sample-broker
- For the git repository of the server distribution: https://github.com/powertac/server-distribution

6 Installation & Execution Instructions

For this project, you will need to download two separate packages, the PowerTAC server distribution and the PowerTAC sample broker. The packages can run in both Windows and Ubuntu, however, we **highly recommend** that you create an Ubuntu workspace. Here, we will present the installation instructions for **Ubuntu**.

6.1 Installing and running the PowerTAC server

1. You will need to have Java JDK-11 or a newer version for both the server and the broker.

2. Install maven:

```
$ sudo apt update
$ sudo apt install maven
```

3. Download the server distribution by opening a terminal and writing the command:

```
$ git clone https://github.com/powertac/server-distribution.git
```

4. You can now run the server either with a visualizer so you can observe the various metrics during the simulations or via the command line to rapidly execute and debug your code. For detailed instructions, follow this link.

   - **Visualizer**:
     (a) Navigate to the root directory of the server-distribution folder and enter the following command:

     ```
     $ mvn –Pweb2
     ```

     (b) Open a browser and after waiting for a few seconds, you can find the visualizer at the http://localhost:8080

     (c) Sign in by going to "Account" − > "Sign in" using the **username: admin** and **password: admin**.

(d) Now, you can create your first “Bootstrap game” by going to the “Games” tab and selecting the “ I want to run a bootstrap game...” option (You will need to create a bootstrap file only the first time you install the visualizer).

(e) You only need to enter a game name and then start the simulation.

(f) After the blue bar has loaded, you can head to the “Graphs” tab so you can observe the game stats for both retail and wholesale markets.

(g) After the bootstrap game finishes, you are ready to start running your own simulations. To do that, go to the “Games” tab and select the “I want to run a simulation ...” option.

(h) Select a game name and then add the name of the brokers you wish to participate in the game. Finally, select the bootstrap file you created earlier and run the game.

(i) Now, the server will wait for all the brokers to connect and then automatically start the game.

(j) You will need to start each broker manually.

- Command Line:

  (a) Run the Visualizer method the first time so you can create a bootstrap file.

  (b) Run the following command:

  $ mvn -Pcli -Dexec.args=”–sim –boot-data bootname –brokers a,b,c”

  , where **bootname** will be the path to the bootstrap file and **a,b,c** will be the name of the brokers you want to participate in, separated with a comma.

6.2 Installing and running the PowerTAC sample broker

1. Download the sample-broker in a separate folder:

   $ git clone https://github.com/powertac/sample-broker.git

2. Import the project to an IDE supporting Spring tools such as Eclipse.

3. Install the spring tools. For the Eclipse IDE, you can open Eclipse’s menu bar and go to “Help” − > “Eclipse Marketplace...”, then enter the keyword “spring” and select “Spring Tools (Standalone Edition)” to install.

4. You can now import the sample-broker project as a maven project by going to the “File” − > “Import” − > “Maven” − > “Existing maven projects”.spring

5. Navigate to the “src//main//java//org.powertac.samplebroker// ” so you can find the two java files you will write code in, “PortfolioManagerService.java” and “MarketManagerService.java”.

6. You can set your broker's name by opening the "broker.properties.pom" file located at the root folder and modifying the corresponding line "samplebroker.core.powerTacBroker.username".
7. To finally run the broker, in Eclipse, right-click the project $->$ Run as application $->$ Search for the "BrokerMain" file and run it.
8. If a server game was already created, your broker will connect to it. (You will need to set up the server games in advance and then run the brokers)

6.3 Running the provided PowerTAC brokers

We will give you the executable files of some PowerTAC brokers so you can run them together with your own broker and compare them. You will also need to provide some metrics for every agent in your final report. You can find the files on the course's website.

6.3.1 Sample Broker

The first executable we give you is a compiled version of the sample broker. You can run it by navigating to the folder "sample.broker.executable" and executing the command:

```
$ java -jar sample-broker.jar
```

The name of the broker is "sb1". You can change the name of the agent running by modifying the broker.properties files in the folder.

6.3.2 TUC-TAC 2020

You can compete against the broker that won PowerTAC 2020. You can run it by navigating to the folder "TUC_TAC" and executing the command:

```
$ java -jar TUC_TAC_2020.jar
```

The name of the broker is "TUC_TAC".

7 Final Deliverable

Each team must submit, a compressed **zip file** containing:

1. The **code of the whole project** in which they developed their broker
2. An **eight-page report**

Good luck!!!